Common Criteria for Information Technology Security Evaluation

CCEB-96/013_D

Part 3: Annex D Evaluation assurance levels

Version 1.00

96/01/31

Foreword

Following extensive international cooperation to align the source criteria from Canada (CTCPEC), Europe (ITSEC) and the United States of America (TCSEC and Federal Criteria), version 1.0 of the *Common Criteria for Information Technology Security Evaluation* is issued for the purpose of trial evaluations and for review by the international security community. The practical experience acquired through trial evaluations and all the comments received will be used to further develop the criteria.

A template for reporting observations on version 1.0 of the CC is included at the end of this document. Any observation reports should be communicated to one or more of the following points of contact at the sponsoring organisations:

National Institute of Standards and Technology
Computer Security Division
NIST North Building, Room 426
Gaithersburg, Maryland 20899
U.S.A.
Tel: (+1)(301)975-2934, Fax:(+1)(301)926-2733
E-mail:csd@nist.gov
http://csrc.ncsl.nist.gov

National Security Agency
Attn: V2, Common Criteria Technical Advisor
Fort George G. Meade, Maryland 21122
U.S.A.
Tel: (+1)(410)859-4458, Fax:(+1)(410)684-7512
E-mail: common_criteria@radium.ncsc.mil

Communications Security Establishment
Criteria Coordinator
R2B IT Security Standards and Initiatives
P.O. Box 9703, Terminal
Ottawa, Canada K1G 3Z4
Tel:(+1)(613)991-7409, Fax:(+1)(613)991-7411
E-mail:criteria@cse.dnd.ca
ftp:ftp.cse.dnd.ca
http://www.cse.dnd.ca

UK IT Security and Certification Scheme
Senior Executive
P.O. Box 152
Cheltenham GL52 5UF
United Kingdom
Tel: (+44) 1242 235739, Fax:(+44)1242 235233
E-mail: ccv1.0@itsec.gov.uk
ftp: ftp.itsec.gov.uk
http://www.itsec.gov.uk

Bundesamt für Sicherheit in der Informationstechnik
Abteilung V
Postfach 20 03 63
D-53133 Bonn
Germany
Tel: (+49)228 9582 300, Fax:(+49)228 9582 427
E-mail:cc@bsi.de

Service Central de la Sécurité des Systèmes d'Information
Bureau Normalisation, Critères Communs
18 rue du docteur Zamenhof
92131 Issy les Moulineaux
France
Tel: (+33)(1)41463784, Fax:(+33)(1)41463701
E-mail:ssi28@calvacom.fr

Netherlands National Communications Security Agency
P.O. Box 20061
NL 2500 EB The Hague
The Netherlands
Tel: (+31).70.3485637, Fax:(+31).70.3486503
E-mail: criteria@nlncsa.minbuza.nl

This document is paginated from i to xii and from 1 to 232

Table of contents

List of tables

Chapter 1

Introduction

1 Part 3 of the CC defines assurance requirements in a hierarchical organisational structure (i.e., classes, families, components, and elements), and groups those requirements into Evaluation Assurance Levels (EALs) by reference. This document defines the assurance requirements for each EAL explicitly.

2 Chapter 2 of this document summarises the EALs in a manner similar to the EAL definitions in Part 3 of the CC. Chapter 3 goes on to represent, for each EAL, the set of objectives, application notes, dependencies, and requirements in the classes, families, and components that are included in the EAL.

3 This annex has been compiled almost exclusively by cross reference to the main body of Part 3. In the event of errors which may have occurred with the cross referencing, the component definitions in the main body take precendence.

Chapter 2

Assurance levels

4 The Evaluation Assurance Levels (EALs) provide a uniformly increasing scale which balances the level of assurance obtained with the cost and feasibility of acquiring that degree of assurance.

5 While the CC has adopted the evaluation-based criteria philosophy of its predecessors, the EALs were developed within that philosophy but with a different scope. The CC approach divides the concepts of assurance in a TOE at the end of the evaluation and maintenance of that assurance during the operational use of the TOE. The result being a departure from the evaluation levels of the various predecessors of the CC inasmuch as some of the assurance families are not included in any EAL.

6 In defining the EALs, an analysis was performed which concluded that every assurance family, except “Delivery” and “Flaw remediation”, contributes directly to the assurance that a TOE meets its security claims at the end of the evaluation. As the assurance paradigm is based on assurance gained during evaluation, the EALs are based on those assurance families. This is supported by the fact that evaluators gain “real” assurance by the first hand application of assurance mechanisms (e.g., analysis and testing of an existing design), while they can gain only “theoretical” assurance for mechanisms applied after the evaluation (e.g., a plan for delivery of the TOE). In other words, while such assurance mechanisms can be evaluated to determine whether they can provide their claimed assurance, it is not possible to produce practical evidence of their future application.

7 It is important to note that the “Delivery” and “Flaw remediation” families, as well as some aspects of the other families (e.g., “CM capabilities”), can be evaluated and provide meaningful and desired assurances. The assurance that they provide contributes to maintaining that initial assurance determined by the evaluation of the TOE. Note that while these families are not specifically included in any EAL, it is expected and recommended that they be considered for augmentation of an EAL in PPs and STs.

2.1 Evaluation assurance level (EAL) overview

Assurance Class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Configuration management	ACM_AUT				**1**	1	**2**	2
	ACM_CAP	**1**	1	**2**	**3**	3	**4**	4
	ACM_SCP			**1**	**2**	**3**	3	3
Delivery and operation	ADO_DEL							
	ADO_IGS		**1**	1	1	1	1	1
Development	ADV_FSP	**1**	1	1	**2**	**4**	**5**	**6**
	ADV_HLD		**1**	**2**	2	**3**	**4**	**5**
	ADV_IMP				**1**	**2**	**3**	3
	ADV_INT					**1**	**2**	**3**
	ADV_LLD				**1**	1	**2**	2
	ADV_RCR	**1**	1	1	1	**2**	2	**3**
Guidance documents	AGD_ADM	**1**	1	1	1	1	1	1
	AGD_USR	**1**	1	1	1	1	1	1
Life cycle support	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD				**1**	**2**	2	**3**
	ALC_TAT				**1**	**2**	**3**	3
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT		**1**	**2**	2	**3**	3	**4**
	ATE_FUN		**1**	1	1	1	1	1
	ATE_IND	**1**	1	**2**	2	2	2	**3**
Vulnerability assessment	AVA_CCA					**1**	**2**	2
	AVA_MSU			**1**	**2**	2	2	2
	AVA_SOF		**1**	1	1	1	1	1
	AVA_VLA		**1**	1	**2**	**3**	**4**	4

Table 2.1 -Evaluation Assurance Level Summary

8 Table 2.1 represents a summary of the EALs. The columns represent a hierarchically ordered set of EALs, while the rows represent assurance families. Each point in the resulting matrix identifies a specific assurance component where applicable.

9 As outlined in the next section, seven hierarchically ordered evaluation assurance levels that can be selected are defined in this CC for the rating of the TOE's assurance. They are hierarchically ordered inasmuch as each EAL represents more assurance than all lower EALs. The increase in assurance from EAL to EAL is

accomplished by *substituting* a hierarchically higher assurance component from the same assurance family (i.e., increasing rigour, scope, and/or depth) and from the *addition* of assurance components from other assurance families (i.e., adding new requirements).

10 These EALs consist of an appropriate combination of assurance components as described in Chapter 2 of this Part. More precisely, each EAL includes no more than one component of each assurance family and all assurance dependencies of every component are addressed.

11 While the EALs are defined in the CC, it is possible to represent other combinations of assurance. Specifically, the notion of "augmentation" allows the addition of assurance components (from assurance families not already included in the EAL) or the substitution of assurance components (with another hierarchically higher assurance component in the same assurance family) to an EAL. Of the assurance constructs defined in the CC, only EALs may be augmented. Furthermore, an EAL may be altered only be augmentation. The notion of an "EAL minus a constituent assurance component" is not recognised by the CC as a valid claim. Augmentation carries with it the obligation on the part of the claimant to justify the utility and added value of the added assurance component to the EAL.

2.2 Evaluation assurance level details

12 The following sections provide definitions of the EALs, highlighting differences between the specific requirements and the prose characterisations of those requirements using bold type.

2.2.1 Evaluation assurance level 1 (EAL1) - functionally tested

Objectives

13 EAL1 is the lowest assurance level for which evaluation is meaningful and economically justified. EAL1 is intended to detect obvious errors for a minimum outlay but is unlikely to result in the detection of other than very obvious security weaknesses.

14 EAL1 is applicable in circumstances where those responsible for user data may wish or be obliged to seek independent assurances in the IT security but the risks to security are not viewed as serious. Under these circumstances, an EAL1 rating would be of value to support the contention that due care had been exercised with respect to personal or similar information.

Assurance components

15 **EAL1 (see Table 2.2) provides a minimum level of assurance by an analysis of the security functions using a functional and interface specification of the TOE, to understand the security behaviour.**

16 **The analysis is supported by independent testing of each of the security functions.**

17 This EAL, nonetheless, represents a meaningful increase over an un-evaluated IT product or system (TOE).

Assurance class	Assurance components
Configuration management	**ACM_CAP.1 Minimal support**
Development	**ADV_FSP.1 TOE and security policy**
	ADV_RCR.1 Informal correspondence demonstration
Guidance documents	**AGD_ADM.1 Administrator guidance**
	AGD_USR.1 User guidance
Tests	**ATE_IND.1 Independent testing - conformance**

Table 2.2 -EAL1

2.2.2 Evaluation assurance level 2 (EAL2) - structurally tested

Objectives

18 EAL2 is the highest assurance level that can be used without imposing other than minimal additional tasks upon the developer. If the developer applies reasonable standards of care to the development, EAL2 may be feasible without developer involvement other than support for security functional testing.

19 EAL2 is therefore applicable in those circumstances where developers or users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record. Such a situation may arise when securing legacy systems or where access to the developer may be limited.

Assurance components

20 **EAL2** (see Table **2.3**) provides assurance by an analysis of the security functions using a functional and interface specification **and the high-level design of the subsystems** of the TOE, to understand the security behaviour**.**

21 The analysis is supported by independent testing of each of the security functions**, evidence of developer "black box" testing, and evidence of a developer search for obvious vulnerabilities (e.g., those in the public domain)**.

22 This EAL represents a meaningful increase in assurance from EAL1 by requiring developer testing, a vulnerability analysis, and independent testing based upon more detailed TOE specifications.

Assurance class	Assurance components
Configuration management	ACM_CAP.1 Minimal support
Delivery and operation	**ADO_IGS.1 Installation, generation, and start-up procedures**
Development	ADV_FSP.1 TOE and security policy
	ADV_HLD.1 Descriptive high-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Tests	**ATE_COV.1 Complete coverage - informal**
	ATE_DPT.1 Testing - functional specification
	ATE_FUN.1 Functional testing
	ATE_IND.1 Independent testing - conformance
Vulnerability assessment	**AVA_SOF.1 Strength of TOE security function evaluation**
	AVA_VLA.1 Developer vulnerability analysis

Table 2.3 -EAL2

2.2.3 Evaluation assurance level 3 (EAL3) - methodically tested and checked

Objectives

23 EAL3 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practices.

24 EAL3 is therefore applicable in those circumstances where developers or users require a moderate level of independently assured security and require a thorough investigation of the product and its development without incurring substantial re-engineering costs.

Assurance components

25 **EAL3** (see Table **2.4**) provides assurance by an analysis of the security functions using a functional and interface specification and the high-level design of the subsystems of the TOE, to understand the security behaviour**.**

26 The analysis is supported by independent testing of the security functions, evidence of developer "**gray** box" testing, **selective independent confirmation of the developer test results**, and evidence of a developer search for obvious vulnerabilities (e.g., those in the public domain).

27 **EAL3 also provides added assurance through the addition of development environment controls and TOE configuration management.**

28 This EAL represents a meaningful increase in assurance from EAL2 by requiring more complete testing coverage of the security functions and mechanisms and/or procedures that provide some confidence that the TOE will not be tampered with during development.

Assurance class	Assurance components
Configuration management	**ACM_CAP.2 Authorisation controls**
	ACM_SCP.1 Minimal CM coverage
Delivery and operation	ADO_IGS.1 Installation, generation, and start-up procedures
Development	ADV_FSP.1 TOE and security policy
	ADV_HLD.2 Security enforcing high-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	**ALC_DVS.1 Identification of security measures**
Tests	**ATE_COV.2 Complete coverage - rigorous**
	ATE_DPT.2 Testing - high level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_MSU.1 Misuse analysis - obvious flaws**
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.1 Developer vulnerability analysis

Table 2.4 -EAL3

2.2.4 Evaluation assurance level 4 (EAL4) - methodically designed, tested, and reviewed

Objectives

29 EAL4 permits a developer to gain maximum assurance from positive security engineering based on good commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources. EAL4 is the highest level which it is likely to be economically feasible to retrofit to an existing product line.

30 EAL4 is therefore applicable in those circumstances where developers or users require a moderate to high level of independently assured security in conventional commodity products and are prepared to incur additional security specific engineering costs.

Assurance components

31 **EAL4** (see Table **2.5**) provides assurance by an analysis of the security functions using a functional and interface specification**,** the high-level design of the subsystems, **the low-level design of the modules of the TOE, and a subset of the implementation**, to understand the security behaviour**.**

32 The analysis is supported by independent testing of the security functions, evidence of developer "gray box" testing, selective independent confirmation of the developer test results, evidence of a developer search for obvious vulnerabilities (e.g., those in the public domain), **and an independent search for obvious vulnerabilities**.

33 **EAL4** also provides assurance through the **use** of development environment controls and **additional** TOE configuration management **including automation**.

34 This EAL represents a meaningful increase in assurance from EAL3 by requiring more design description, a subset of the implementation, and improved mechanisms and/or procedures that provide confidence that the TOE will not be tampered with during development.

Assurance class	Assurance components
Configuration management	**ACM_AUT.1 Partial CM automation**
	ACM_CAP.3 Generation support and acceptance procedures
	ACM_SCP.2 Problem tracking CM coverage
Delivery and operation	ADO_IGS.1 Installation, generation, and start-up procedures
Development	**ADV_FSP.2 Informal security policy model**
	ADV_HLD.2 Security enforcing high-level design
	ADV_IMP.1 Subset of the implementation of the TSF
	ADV_LLD.1 Descriptive low-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	**ALC_DVS.1 Identification of security measures**
	ALC_LCD.1 Developer defined life-cycle model
	ALC_TAT.1 Well defined development tools
Tests	ATE_COV.2 Complete coverage - rigorous
	ATE_DPT.2 Testing - high level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_MSU.2 Misuse analysis - independent verification**
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.2 Independent vulnerability analysis

Table 2.5 -EAL4

2.2.5 Evaluation assurance level 5 (EAL5) - semiformally designed and tested

Objectives

35 EAL5 permits a developer to gain maximum assurance from security engineering based upon rigorous commercial development practices supported by moderate application of specialist security engineering techniques. Such a product will be designed and developed with the intent of achieving EAL5 assurance. It is likely that the additional costs attributable to the EAL5 requirements relative to rigorous development without the application of specialised techniques will not be excessive.

36 EAL5 is therefore applicable in those circumstances where developers or users require a high level of independently assured security in a planned development and require a rigorous development approach without incurring unreasonable costs attributable to specialist security engineering techniques.

Assurance components

37 **EAL5** (see Table **2.6**) provides assurance by an analysis of the security functions using a functional and interface specification**,** the high-level design of the subsystems, the low-level design of the modules of the TOE, and **all** of the implementation, to understand the security behaviour**. Assurance is additionally gained through a formal model and a semiformal presentation of the functional specification and high-level design and a semiformal demonstration of correspondence between them.**

38 The analysis is supported by independent testing of the security functions, evidence of developer "gray box" testing, selective independent confirmation of the developer test results, evidence of a developer search for obvious vulnerabilities (e.g., those in the public domain), and an independent search for **vulnerabilities ensuring relative resistance to penetration attack**. **The analysis also includes a search for covert channels, when applicable, and is supported by requiring a modular TOE design.**

39 **EAL5** also provides assurance through the use of a development environment controls, and **comprehensive** TOE configuration management including automation.

40 This EAL represents a meaningful increase in assurance from EAL4 by requiring semiformal design descriptions, the entire implementation, a more structured (and hence analysable) architecture, covert channel analysis, and improved mechanisms and/or procedures that provide confidence that the TOE will not be tampered with during development.

Assurance class	Assurance components
Configuration management	ACM_AUT.1 Partial CM automation
	ACM_CAP.3 Generation support and acceptance procedures
	ACM_SCP.3 Development tools CM coverage
Delivery and operation	ADO_IGS.1 Installation, generation, and start-up procedures
Development	**ADV_FSP.4 Formal security policy model**
	ADV_HLD.3 Semiformal high-level design
	ADV_IMP.2 Implementation of the TSF
	ADV_INT.1 Modularity
	ADV_LLD.1 Descriptive low-level design
	ADV_RCR.2 Semiformal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	ALC_DVS.1 Identification of security measures
	ALC_LCD.2 Standardised life-cycle model
	ALC_TAT.2 Compliance with implementation standards
Tests	ATE_COV.2 Complete coverage - rigorous
	ATE_DPT.3 Testing - low level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_CCA.1 Covert channel analysis**
	AVA_MSU.2 Misuse analysis - independent verification
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.3 Relatively resistant

Table 2.6 -EAL5

2.2.6 Evaluation assurance level 6 (EAL6) - semiformally verified design and tested

Objectives

41 EAL6 permits developers to gain high assurance from application of security engineering techniques to a rigorous development environment in order to produce a premium product for protecting high value assets against significant risks.

42 EAL6 is therefore applicable to the development of specialist security products for application in high risk situations where the value of the protected assets justifies the additional costs.

Assurance components

43 **EAL6** (see Table **2.7**) provides assurance by an analysis of the security functions using a functional and interface specification**,** the high-level design of the subsystems, the low-level design of the modules of the TOE, and **a structured presentation** of the implementation, to understand the security behaviour**.** Assurance is additionally gained through a formal model, a semiformal presentation of the functional specification, high-level design**, and low-level design** and a semiformal demonstration of correspondence between them.

44 The analysis is supported by independent testing of the security functions, evidence of developer "gray box" testing, selective independent confirmation of the developer test results, evidence of a developer search for obvious vulnerabilities (e.g., those in the public domain), and an independent search for vulnerabilities ensuring **high** resistance to penetration attack. The analysis also includes a **systematic** search for covert channels, when applicable, and is supported by requiring a modular **and layered** TOE design.

45 **EAL6** also provides assurance through the use of a **structured development process,** development environment controls, and comprehensive TOE configuration management including **complete** automation.

46 This EAL represents a meaningful increase in assurance from EAL5 by requiring more comprehensive analysis, a structured representation of the implementation, more architectural structure (e.g., layering), more comprehensive independent vulnerability analysis, systematic covert channel identification, and improved configuration management and development environment controls.

Assurance class	Assurance components
Configuration management	**ACM_AUT.2 Complete CM automation**
	ACM_CAP.4 Advanced support
	ACM_SCP.3 Development tools CM coverage
Delivery and operation	ADO_IGS.1 Installation, generation, and start-up procedures
Development	**ADV_FSP.5 Property specification by model interpretation**
	ADV_HLD.4 Semiformal high-level explanation
	ADV_IMP.3 Structured implementation of the TSF
	ADV_INT.2 Layering
	ADV_LLD.2 Semiformal low-level design
	ADV_RCR.2 Semiformal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	**ALC_DVS.2 Sufficiency of security measures**
	ALC_LCD.2 Standardised life-cycle model
	ALC_TAT.3 Compliance with implementation standards - all parts
Tests	**ATE_COV.3 Ordered testing**
	ATE_DPT.3 Testing - low level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_CCA.2 Systematic covert channel analysis**
	AVA_MSU.2 Misuse analysis - independent verification
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.4 Highly resistant

Table 2.7 -EAL6

2.2.7 Evaluation assurance level 7 (EAL7) - formally verified design and tested

Objectives

47 EAL7 represents an achievable upper bound on evaluation assurance for practically useful products and should only be considered for experimental application to all but conceptually simple and well understood products.

48 EAL7 is therefore applicable to the development of specialist security products for application in extremely high risk situations and/or where the high value of the assets justifies the higher costs. Practical application of EAL7 is currently limited to products with tightly focused security functionality which is amenable to formal analysis.

Assurance components

49 **EAL7** (see Table **2.8**) provides assurance by an analysis of the security functions using a functional and interface specification, the high-level design of the subsystems, the low-level design of the modules of the TOE, and a structured presentation of the implementation, to understand the security behaviour. Assurance is additionally gained through a formal model**, a formal presentation of the functional specification and high-level design**, a semiformal presentation of the low-level design, and **formal and** semiformal demonstration of correspondence between them**, as appropriate**.

50 The analysis is supported by independent testing of the security functions, evidence of developer "**white** box" testing, **complete** independent confirmation of the developer test results, evidence of a developer search for obvious vulnerabilities (e.g., those in the public domain), and an independent search for vulnerabilities ensuring high resistance to penetration attack. The analysis also includes a systematic search for covert channels, when applicable, and is supported by requiring a modular, layered, **and simple** TOE design.

51 **EAL7** also provides assurance through the use of a structured development process, development environment controls, and comprehensive TOE configuration management including complete automation.

52 This EAL represents a meaningful increase in assurance from EAL6 by requiring more comprehensive analysis using formal representations and formal correspondence, comprehensive testing, and exhaustive covert channel analysis**.**

Assurance class	Assurance components
Configuration management	ACM_AUT.2 Complete CM automation
	ACM_CAP.4 Advanced support
	ACM_SCP.3 Development tools CM coverage
Delivery and operation	ADO_IGS.1 Installation, generation, and start-up procedures
Development	**ADV_FSP.6 Formal specification of the TSF properties**
	ADV_HLD.5 Formal high-level design
	ADV_IMP.3 Structured implementation of the TSF
	ADV_INT.3 Minimisation of Complexity
	ADV_LLD.2 Semiformal low-level design
	ADV_RCR.3 Formal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	ALC_DVS.2 Sufficiency of security measures
	ALC_LCD.3 Measurable life-cycle model
	ALC_TAT.3 Compliance with implementation standards - all parts
Tests	ATE_COV.3 Ordered testing
	ATE_DPT.4 Testing - implementation
	ATE_FUN.1 Functional testing
	ATE_IND.3 Independent testing - complete
Vulnerability assessment	AVA_CCA.2 Systematic covert channel analysis
	AVA_MSU.2 Misuse analysis - independent verification
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.4 Highly resistant

Table 2.8 -EAL7

Chapter 3

Detailed EAL requirements

53 The following sections fully expand the requirements for each EAL. The requirements are exactly as they appear in Part 3 of the CC, except that:

a) only the relevant classes, families, and components are included for each EAL;

b) component levelling is not represented; and

c) requirement highlighting, to indicate differences from the preceding EAL, occurs only at the granularity of an “element”, as opposed to individual word changes.

EAL 1
Functionally tested

ACM Configuration management

54 Configuration management (CM) is an aspect of establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE. CM systems are put in place to ensure the integrity of the configuration items that they control, by providing a method of tracking these configuration items, and by ensuring that only authorised users are capable of changing them.

ACM_CAP CM capabilities

Objectives

55 The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TSF from the early design stages through all subsequent maintenance efforts.

56 The objectives of this family include the following:

a) ensuring that the TSF is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items; and

d) enabling recovery to an earlier version of the TOE, in the event that an error occurs through modification, addition, or deletion of TOE configuration items.

Application notes

57 For ACM_CAP.1 and the higher components, there is a requirement that a configuration list be provided. The configuration list contains all configuration items which are maintained by the CM system.

58 For ACM_CAP.2 and the higher components, there is a requirement that the CM documentation include evidence that the CM system is working properly. An example of such evidence might be audit trail output from the CM system. The

evaluator is responsible for examining such evidence, to determine that it is sufficient to demonstrate proper functionality of the CM system.

59 For ACM_CAP.2 and the higher components, there is a requirement that evidence be provided that all configuration items are being maintained under the CM system. Since a configuration item refers to an item which is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

60 For ACM_CAP.3 and ACM_CAP.4, there is a requirement that the CM system support the generation of all supported versions of the TOE. This provides the ability to recover to a previous known version in the event that an error occurs through modification, addition or deletion of TOE configuration items.

ACM_CAP.1 Minimal support

Objectives

61 Clear identification of the TOE is required to determine those items under evaluation that are subject to the criteria requirements.

Dependencies:

No dependencies.

Developer action elements:

ACM_CAP.1.1D **The developer shall use a CM system.**

ACM_CAP.1.2D **The developer shall provide CM documentation.**

Content and presentation of evidence elements:

ACM_CAP.1.1C **The CM documentation shall include a configuration list.**

ACM_CAP.1.2C **The configuration list shall describe the configuration items that comprise the TOE.**

ACM_CAP.1.3C **The CM documentation shall describe the method used to uniquely identify the TOE configuration items.**

Evaluator action elements:

ACM_CAP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV Development

62 The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. The other family in the development class describes requirements for the internal structure of the TSF.

63 The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, since some of the representations are not necessary for low assurance evaluations.

ADV_FSP Functional specification

Objectives

64 The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is a refinement of the statement of IT functional requirements in the ST of the TOE. The functional specification has to show that all the functional requirements defined in the ST are addressed, and that the TSP is enforced by the TSF.

Application notes

65 In addition to the content indicated in the following requirements, the functional specification shall also include any additional specific detail specified by the documentation notes in the related functional components.

66 The developer must provide evidence that the TSF is completely represented by the functional specification. While a functional specification for the entire TOE would allow an evaluator to determine the TSF boundary, it is not necessary to require that specification when other evidence could be provided to demonstrate the TSF boundary.

67 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the functional specification. In the course of the functional specification evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of

another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

68 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

69 While a TSP may represent any policies, TSP models have traditionally represented only subsets of those policies. As a result, the TSP model cannot be treated like every other TSF representation inasmuch as the correspondence between the TSP model to the adjacent abstractions (i.e., TSP and functional specification) may not be complete. As a result, there must be a demonstration of correspondence from the functional specification to the TSP directly, rather than through the intervening representation (i.e., TSP model) where correspondence may be lost. For these reasons, all of the requirements for correspondence between the TSP, TSP model, and functional specification have been included in this family and the correspondence requirements in the Representation correspondence (ADV_RCR) family do not apply to the TSP and TSP model.

70 Beginning with ADV_FSP.1, requirements are defined to ensure that the functional specification is consistent with the TSP. Beginning with ADV_FSP.2, because there is no requirement for a TSP model in ADV_FSP.1, requirements are defined to describe the rules and characteristics of applicable policies of the TSP in the TSP model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g., state transition, non-interference). For example, rules may be represented as “properties” (e.g., simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects”, and “objects”.

71 Since not all policies can be modeled, given the current state of the art, the requirement indicating which policies shall be modeled is subjective. The PP/ST author should identify specific functions and associated policies that are required to be modeled. At the very least, access control policies are expected to be modeled since they are currently within the state of the art.

ADV_FSP.1 TOE and security policy

Dependencies:

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.1.1D **The developer shall provide a functional specification.**

ADV_FSP.1.2D **The developer shall provide a TSP.**

Content and presentation of evidence elements:

ADV_FSP.1.1C **The functional specification shall describe the TSF using an informal style.**

ADV_FSP.1.2C **The functional specification shall include an informal presentation of syntax and semantics of all external TSF interfaces.**

ADV_FSP.1.3C **The functional specification shall include evidence that demonstrates that the TSF is completely represented.**

Evaluator action elements:

ADV_FSP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_FSP.1.2E **The evaluator shall determine that the functional specification is consistent with the TSP.**

ADV_FSP.1.3E **The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.**

ADV_RCR Representation correspondence

Objectives

72 The correspondence between the various representations (i.e. functional requirements expressed in the ST, functional specification, high-level design, low-level design, implementation) addresses the correct and complete instantiation of the requirements to the least abstract representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

Application notes

73 The developer must demonstrate to the evaluator that the most detailed, or least abstract, representation of the TSF is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

74 The evaluator must analyse each demonstration of correspondence between abstractions, as well as the results of the analysis of each TSF representation, and

then make a determination as to whether the functional requirements in the ST have been satisfied.

75 This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure 5.4, it is intended to address correspondence between the requirements in the ST as well as the TOE summary specification, functional specification, high-level design, low-level design, and implementation representation.

ADV_RCR.1 Informal correspondence demonstration

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.1.1D **The developer shall provide evidence that the least abstract TSF representation provided is an accurate, consistent, and complete instantiation of the functional requirements expressed in the ST.**

Content and presentation of evidence elements:

ADV_RCR.1.1C **For each adjacent pair of TSF representations, the evidence shall demonstrate that all parts of the more abstract representation are refined in the less abstract representation.**

ADV_RCR.1.2C **For each adjacent pair of TSF representations, the demonstration of correspondence between the representations may be informal.**

Evaluator action elements:

ADV_RCR.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_RCR.1.2E **The evaluator shall analyse the correspondence between the functional requirements expressed in the ST and the least abstract representation provided to ensure accuracy, consistency, and completeness.**

AGD Guidance documents

76 The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure installation and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

AGD_ADM Administrator guidance

Objectives

77 Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

Application notes

78 The requirements AGD_ADM.1.2C and AGD_ADM.1.11C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

79 The PP/ST author should review the functional components of the PP/ST for guidance on administrator documentation. Those application notes that are relevant to administrator guidance for understanding and proper application of the security functions should be considered for inclusion in the administrator guidance requirements. An example of an administrator guidance document is a reference manual.

AGD_ADM.1 Administrator guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_ADM.1.1D **The developer shall provide administrator guidance addressed to system administrative personnel.**

Content and presentation of evidence elements:

AGD_ADM.1.1C **The administrator guidance shall describe how to administer the TOE in a secure manner.**

AGD_ADM.1.2C **The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.**

AGD_ADM.1.3C **The administrator guidance shall contain guidelines on the consistent and effective use of the security functions within the TSF.**

AGD_ADM.1.4C **The administrator guidance shall describe the difference between two types of functions: those which allow an administrator to control security parameters, and those which allow the administrator to obtain information only.**

AGD_ADM.1.5C **The administrator guidance shall describe all security parameters under the administrator's control.**

AGD_ADM.1.6C **The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.**

AGD_ADM.1.7C **The administrator guidance shall contain guidelines on how the security functions interact.**

AGD_ADM.1.8C **The administrator guidance shall contain instructions regarding how to configure the TOE.**

AGD_ADM.1.9C **The administrator guidance shall describe all configuration options that may be used during secure installation of the TOE.**

AGD_ADM.1.10C **The administrator guidance shall describe details, sufficient for use, of procedures relevant to the administration of security.**

AGD_ADM.1.11C **The administrator guidance shall be consistent with all other documents supplied for evaluation.**

Evaluator action elements:

AGD_ADM.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AGD_ADM.1.2E **The evaluator shall confirm that the installation procedures result in a secure configuration.**

AGD_USR User guidance

Objectives

80 User guidance refers to written material that is intended to be used by nonadministrative (human) users of the TOE. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

81 The user guidance provides a basis for assumptions about the use of the TOE and a measure of confidence that non-malicious users and application providers will understand the secure operation of the TOE and will use it as intended.

Application notes

82 The requirement AGD_USR.1.3.C and AGD_USR.1.5C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the user guidance.

83 The PP/ST author should review the functional components of the PP/ST for guidance on user documentation. Those application notes that are relevant to user guidance aimed at the understanding and proper use of the security functions should be considered for inclusion in the user guidance requirements. Examples of user guidance are reference manuals, user guides, and on-line help.

AGD_USR.1 User guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_USR.1.1D **The developer shall provide user guidance.**

Content and presentation of evidence elements:

AGD_USR.1.1C **The user guidance shall describe the TSF and interfaces available to the user.**

AGD_USR.1.2C **The user guidance shall contain guidelines on the use of security functions provided by the TOE.**

AGD_USR.1.3C **The user guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.**

AGD_USR.1.4C **The user guidance shall describe the interaction between user-visible security functions.**

AGD_USR.1.5C **The user guidance shall be consistent with all other documentation delivered for evaluation.**

Evaluator action elements:

AGD_USR.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE Tests

84 The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g., functional testing performed by evaluators)

(ATE_IND), and functional tests (ATE_FUN). Testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of the PP/ ST. Testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. Testing may also be directed toward the internals of the TSF, such as the testing of subsystems and modules against their specifications.

85 The aspects of coverage and depth have been separated from functional tests for reasons of increased flexibility in applying the components of the families. However, the requirements in these three families are intended to be applied together.

86 The independent testing has dependencies on the other families to provide the necessary information to support the requirements, but is primarily concerned with independent evaluator actions.

87 This class does not address penetration testing, which is directed toward finding vulnerabilities that enable a user to violate the security policy. Penetration testing is addressed separately as an aspect of vulnerability assessment in the class AVA.

ATE_IND Independent testing

Objectives

88 The objective is to demonstrate that the security functions perform as specified.

89 Additionally, an objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer which results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

Application notes

90 The testing specified in this family can be performed by a party other than the evaluator (e.g., an independent laboratory, an objective consumer organisation).

91 This family deals with the degree to which there is independent functional testing of the TOE. Independent functional testing may take the form of repeating the developer's functional tests, in whole or in part. It may also take the form of the augmentation of the developer's functional tests, either to extend the scope or the depth of the developer's tests.

ATE_IND.1 Independent testing - conformance

Objectives

92 In this component, the objective is to demonstrate that the security functions perform as specified.

Application notes

93 The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required to run tests. The need for documentation is supported by the dependencies to other assurance families.

94 Additionally, suitability of the TOE for testing may be based on other considerations e.g., the version of the TOE submitted by the developer is not the final version.

Dependencies:

ADV_FSP.1 TOE and security policy

AGD_USR.1 User guidance

AGD_ADM.1 Administrator guidance

Developer action elements:

ATE_IND.1.1D **The developer shall provide the TOE for testing.**

Content and presentation of evidence elements:

ATE_IND.1.1C **The TOE shall be suitable for testing.**

Evaluator action elements:

ATE_IND.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE_IND.1.2E **The evaluator shall test the TSF to confirm that the TSF operates as specified.**

EAL 2
Structurally tested

ACM Configuration management

95 Configuration management (CM) is an aspect of establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE. CM systems are put in place to ensure the integrity of the configuration items that they control, by providing a method of tracking these configuration items, and by ensuring that only authorised users are capable of changing them.

ACM_CAP CM capabilities

Objectives

96 The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TSF from the early design stages through all subsequent maintenance efforts.

97 The objectives of this family include the following:

a) ensuring that the TSF is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items; and

d) enabling recovery to an earlier version of the TOE, in the event that an error occurs through modification, addition, or deletion of TOE configuration items.

Application notes

98 For ACM_CAP.1 and the higher components, there is a requirement that a configuration list be provided. The configuration list contains all configuration items which are maintained by the CM system.

99 For ACM_CAP.2 and the higher components, there is a requirement that the CM documentation include evidence that the CM system is working properly. An example of such evidence might be audit trail output from the CM system. The

evaluator is responsible for examining such evidence, to determine that it is sufficient to demonstrate proper functionality of the CM system.

100 For ACM_CAP.2 and the higher components, there is a requirement that evidence be provided that all configuration items are being maintained under the CM system. Since a configuration item refers to an item which is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

101 For ACM_CAP.3 and ACM_CAP.4, there is a requirement that the CM system support the generation of all supported versions of the TOE. This provides the ability to recover to a previous known version in the event that an error occurs through modification, addition or deletion of TOE configuration items.

ACM_CAP.1 Minimal support

Objectives

102 Clear identification of the TOE is required to determine those items under evaluation that are subject to the criteria requirements.

Dependencies:

No dependencies.

Developer action elements:

ACM_CAP.1.1D The developer shall use a CM system.

ACM_CAP.1.2D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.1.1C The CM documentation shall include a configuration list.

ACM_CAP.1.2C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.1.3C The CM documentation shall describe the method used to uniquely identify the TOE configuration items.

Evaluator action elements:

ACM_CAP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO Delivery and operation

103 Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

ADO_IGS Installation, generation, and start-up

Objectives

104 Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started in a secure manner as intended by the developer.

Application notes

105 The generation requirements are applicable only to TOEs that provide the ability to generate an operational TOE from source or object code.

106 The installation, generation, and start-up procedures may exist as a separate document, but would typically be grouped with other administrative guidance.

ADO_IGS.1 Installation, generation, and start-up procedures

Dependencies:

AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.1.1D **The developer shall document procedures to be used for the secure installation, generation, and start-up of the TOE.**

Content and presentation of evidence elements:

ADO_IGS.1.1C **The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.**

Evaluator action elements:

ADO_IGS.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV Development

107 The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation. The development class also includes a family of requirements for

a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. The other family in the development class describes requirements for the internal structure of the TSF.

108 The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, since some of the representations are not necessary for low assurance evaluations.

ADV_FSP Functional specification

Objectives

109 The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is a refinement of the statement of IT functional requirements in the ST of the TOE. The functional specification has to show that all the functional requirements defined in the ST are addressed, and that the TSP is enforced by the TSF.

Application notes

110 In addition to the content indicated in the following requirements, the functional specification shall also include any additional specific detail specified by the documentation notes in the related functional components.

111 The developer must provide evidence that the TSF is completely represented by the functional specification. While a functional specification for the entire TOE would allow an evaluator to determine the TSF boundary, it is not necessary to require that specification when other evidence could be provided to demonstrate the TSF boundary.

112 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the functional specification. In the course of the functional specification evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

113 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

114 While a TSP may represent any policies, TSP models have traditionally represented only subsets of those policies. As a result, the TSP model cannot be treated like every other TSF representation inasmuch as the correspondence between the TSP model to the adjacent abstractions (i.e., TSP and functional specification) may not be complete. As a result, there must be a demonstration of correspondence from the functional specification to the TSP directly, rather than through the intervening representation (i.e., TSP model) where correspondence may be lost. For these reasons, all of the requirements for correspondence between the TSP, TSP model, and functional specification have been included in this family and the correspondence requirements in the Representation correspondence (ADV_RCR) family do not apply to the TSP and TSP model.

115 Beginning with ADV_FSP.1, requirements are defined to ensure that the functional specification is consistent with the TSP. Beginning with ADV_FSP.2, because there is no requirement for a TSP model in ADV_FSP.1, requirements are defined to describe the rules and characteristics of applicable policies of the TSP in the TSP model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g., state transition, non-interference). For example, rules may be represented as “properties” (e.g., simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects”, and “objects”.

116 Since not all policies can be modeled, given the current state of the art, the requirement indicating which policies shall be modeled is subjective. The PP/ST author should identify specific functions and associated policies that are required to be modeled. At the very least, access control policies are expected to be modeled since they are currently within the state of the art.

ADV_FSP.1 TOE and security policy

Dependencies:

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.1.1D The developer shall provide a functional specification.

ADV_FSP.1.2D The developer shall provide a TSP.

Content and presentation of evidence elements:

ADV_FSP.1.1C The functional specification shall describe the TSF using an informal style.

ADV_FSP.1.2C The functional specification shall include an informal presentation of syntax and semantics of all external TSF interfaces.

ADV_FSP.1.3C The functional specification shall include evidence that demonstrates that the TSF is completely represented.

Evaluator action elements:

ADV_FSP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2E The evaluator shall determine that the functional specification is consistent with the TSP.

ADV_FSP.1.3E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_HLD High-level design

Objectives

117 The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e., subsystems) and relates these units to the functions that they contain. The high-level design provides assurance that the TOE provides an architecture appropriate to implement the claimed functional requirements.

118 The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function and identifies the security functions enforced by the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

Application notes

119 In addition to the content indicated in the following requirements, the high-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

120 The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

121 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the high-level design. In the course of the high-level design evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

122 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

123 The term “security functionality” is used to represent operations that a subsystem performs that have some effect on the security functions implemented by the TOE. This distinction is made because design constructs, such as subsystems and modules, do not necessarily relate to specific security functions. While a given subsystem may correspond directly to a security function, or even multiple security functions, it is also possible that many subsystems must be combined to implement a single security function.

124 The term “TSP enforcing subsystems” refers to a subsystem that contributes to the enforcement of the TSP.

ADV_HLD.1 Descriptive high-level design

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_HLD.1.1D **The developer shall provide the high-level design of the TSF.**

Content and presentation of evidence elements:

ADV_HLD.1.1C **The presentation of the high-level design shall be informal.**

ADV_HLD.1.2C **The high-level design shall describe the structure of the TSF in terms of subsystems.**

ADV_HLD.1.3C **The high-level design shall describe the security functionality provided by each subsystem of the TSF.**

ADV_HLD.1.4C **The high-level design shall identify the interfaces of the subsystems of the TSF.**

ADV_HLD.1.5C **The high-level design shall identify any underlying hardware, firmware, and/ or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.**

Evaluator action elements:

ADV_HLD.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_HLD.1.2E **The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.**

ADV_RCR Representation correspondence

Objectives

125 The correspondence between the various representations (i.e. functional requirements expressed in the ST, functional specification, high-level design, low-level design, implementation) addresses the correct and complete instantiation of the requirements to the least abstract representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

Application notes

126 The developer must demonstrate to the evaluator that the most detailed, or least abstract, representation of the TSF is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

127 The evaluator must analyse each demonstration of correspondence between abstractions, as well as the results of the analysis of each TSF representation, and then make a determination as to whether the functional requirements in the ST have been satisfied.

128 This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure 5.4, it is intended to address correspondence between the requirements in the ST as well as the TOE summary specification, functional specification, high-level design, low-level design, and implementation representation.

ADV_RCR.1 Informal correspondence demonstration

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.1.1D The developer shall provide evidence that the least abstract TSF representation provided is an accurate, consistent, and complete instantiation of the functional requirements expressed in the ST.

Content and presentation of evidence elements:

ADV_RCR.1.1C For each adjacent pair of TSF representations, the evidence shall demonstrate that all parts of the more abstract representation are refined in the less abstract representation.

ADV_RCR.1.2C For each adjacent pair of TSF representations, the demonstration of correspondence between the representations may be informal.

Evaluator action elements:

ADV_RCR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.1.2E The evaluator shall analyse the correspondence between the functional requirements expressed in the ST and the least abstract representation provided to ensure accuracy, consistency, and completeness.

AGD Guidance documents

129 The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure installation and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

AGD_ADM Administrator guidance

Objectives

130 Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform

security-critical actions and those functions that provide security-critical information.

Application notes

131 The requirements AGD_ADM.1.2C and AGD_ADM.1.11C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

132 The PP/ST author should review the functional components of the PP/ST for guidance on administrator documentation. Those application notes that are relevant to administrator guidance for understanding and proper application of the security functions should be considered for inclusion in the administrator guidance requirements. An example of an administrator guidance document is a reference manual.

AGD_ADM.1 Administrator guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.2C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.3C The administrator guidance shall contain guidelines on the consistent and effective use of the security functions within the TSF.

AGD_ADM.1.4C The administrator guidance shall describe the difference between two types of functions: those which allow an administrator to control security parameters, and those which allow the administrator to obtain information only.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the administrator's control.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall contain guidelines on how the security functions interact.

AGD_ADM.1.8C The administrator guidance shall contain instructions regarding how to configure the TOE.

AGD_ADM.1.9C The administrator guidance shall describe all configuration options that may be used during secure installation of the TOE.

AGD_ADM.1.10C The administrator guidance shall describe details, sufficient for use, of procedures relevant to the administration of security.

AGD_ADM.1.11C The administrator guidance shall be consistent with all other documents supplied for evaluation.

Evaluator action elements:

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_ADM.1.2E The evaluator shall confirm that the installation procedures result in a secure configuration.

AGD_USR User guidance

Objectives

133 User guidance refers to written material that is intended to be used by nonadministrative (human) users of the TOE. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

134 The user guidance provides a basis for assumptions about the use of the TOE and a measure of confidence that non-malicious users and application providers will understand the secure operation of the TOE and will use it as intended.

Application notes

135 The requirement AGD_USR.1.3.C and AGD_USR.1.5C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the user guidance.

136 The PP/ST author should review the functional components of the PP/ST for guidance on user documentation. Those application notes that are relevant to user guidance aimed at the understanding and proper use of the security functions should be considered for inclusion in the user guidance requirements. Examples of user guidance are reference manuals, user guides, and on-line help.

AGD_USR.1 User guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the TSF and interfaces available to the user.

AGD_USR.1.2C The user guidance shall contain guidelines on the use of security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall describe the interaction between user-visible security functions.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation delivered for evaluation.

Evaluator action elements:

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE Tests

137 The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g., functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of the PP/ ST. Testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. Testing may also be directed toward the internals of the TSF, such as the testing of subsystems and modules against their specifications.

138 The aspects of coverage and depth have been separated from functional tests for reasons of increased flexibility in applying the components of the families. However, the requirements in these three families are intended to be applied together.

139 The independent testing has dependencies on the other families to provide the necessary information to support the requirements, but is primarily concerned with independent evaluator actions.

140 This class does not address penetration testing, which is directed toward finding vulnerabilities that enable a user to violate the security policy. Penetration testing is addressed separately as an aspect of vulnerability assessment in the class AVA.

ATE_COV Coverage

Objectives

141 This family addresses those aspects of testing that deal with completeness of testing. That is, it addresses the extent to which the TOE security functions are tested, whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified, and whether or not the order in which testing proceeds correctly accounts for functional dependencies between the portions of the TOE being tested.

Application notes

142 The specific documentation required by the coverage components will be determined, in most cases, by the documentation stipulated in the level of ATE_FUN that is specified. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_COV.1 Complete coverage - informal

Objectives

143 In this component, the objective is that testing completely address the security functions.

Application notes

144 While the testing objective is to completely cover the TSF, there is no more than informal explanation to support this assertion.

Dependencies:

ADV_FSP.1 TOE and security policy

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.1.1D **The developer shall provide an analysis of the test coverage.**

Content and presentation of evidence elements:

ATE_COV.1.1C **The analysis of the test coverage shall demonstrate that the tests identified in the test documentation cover the TSF.**

Evaluator action elements:

ATE_COV.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE_DPT Depth

Objectives

145 The components in this family deal with the level of detail to which the TOE is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.

146 The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internals of the TOE, are more likely to discover any malicious code that has been inserted.

Application notes

147 The specific amount and type of documentation and evidence will, in general, be determined by that required by level of ATE_FUN selected. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_DPT.1 Testing - functional specification

Objectives

148 The functional specification of a TOE provides a high level description of the external workings of the TSF. Testing at the level of the functional specification, in order to demonstrate the presence of any flaws, provides assurance that the TSF functional specification has been correctly realised.

Application notes

149 The functional specification representation is used to express the notion of the most abstract representation of the TSF.

Dependencies:

ADV_FSP.1 TOE and security policy

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.1.1D **The developer shall provide the analysis of the depth of testing.**

Content and presentation of evidence elements:

ATE_DPT.1.1C **The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TOE operates in accordance with the functional specification of the TSF.**

Evaluator action elements:

ATE_DPT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE_FUN Functional tests

Objectives

150 Functional testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family “Functional tests” is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through testing.

151 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

Application notes

152 Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures should also show how the test results is derived from the test inputs.

153 The developer shall eliminate all security relevant flaws discovered during testing.

154 The developer shall test the TSF to determine that no new security relevant flaws have been introduced as a result of eliminating discovered security relevant flaws.

ATE_FUN.1 Functional testing

Objectives

155 The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

Dependencies:

ATE_COV.1 Complete coverage - informal

ATE_DPT.1 Testing - functional specification

Developer action elements:

ATE_FUN.1.1D **The developer shall test the TSF and document the results.**

ATE_FUN.1.2D **The developer shall provide test documentation.**

Content and presentation of evidence elements:

ATE_FUN.1.1C **The test documentation shall consist of test plans, test procedure descriptions, and test results.**

ATE_FUN.1.2C **The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.**

ATE_FUN.1.3C **The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function.**

ATE_FUN.1.4C **The test results in the test documentation shall show the expected results of each test.**

ATE_FUN.1.5C **The test results from the developer execution of the tests shall demonstrate that each security function operates as specified.**

Evaluator action elements:

ATE_FUN.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE_IND Independent testing

Objectives

156 The objective is to demonstrate that the security functions perform as specified.

157 Additionally, an objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer which results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

Application notes

158 The testing specified in this family can be performed by a party other than the evaluator (e.g., an independent laboratory, an objective consumer organisation).

159 This family deals with the degree to which there is independent functional testing of the TOE. Independent functional testing may take the form of repeating the developer's functional tests, in whole or in part. It may also take the form of the augmentation of the developer's functional tests, either to extend the scope or the depth of the developer's tests.

ATE_IND.1 Independent testing - conformance

Objectives

160 In this component, the objective is to demonstrate that the security functions perform as specified.

Application notes

161 The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required to run tests. The need for documentation is supported by the dependencies to other assurance families.

162 Additionally, suitability of the TOE for testing may be based on other considerations e.g., the version of the TOE submitted by the developer is not the final version.

Dependencies:

ADV_FSP.1 TOE and security policy

AGD_USR.1 User guidance

AGD_ADM.1 Administrator guidance

Developer action elements:

ATE_IND.1.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.1.1C The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.2E The evaluator shall test the TSF to confirm that the TSF operates as specified.

AVA Vulnerability assessment

163 The class "Vulnerability assessment" encompasses four families: covert channel analysis (AVA_CCA), misuse (AVA_MSU), strength of TOE security functions

(AVA_SOF) and vulnerability analysis (AVA_VLA). The class addresses the existence of exploitable covert channels, the misuse or incorrect configuration of the TOE, the ability for all critical security mechanisms to withstand direct attack and the definition and assessment of penetration tests to exploit vulnerabilities introduced in the development or the operation of the TOE.

AVA_SOF Strength of TOE security functions

Objectives

164 Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security functions claim.

Application notes

165 Security functions are implemented by security mechanisms. For example, a password mechanism can be used in the implementation of the identification and authentication security function.

166 The strength of TOE security functions evaluation is performed at the level of the security mechanism, but its results provide knowledge about the ability of the related security function to counter the identified threats.

167 The strength of a function is rated ‘basic’ if the analysis shows that the function provides adequate protection against unintended or casual breach of TOE security by attackers possessing a low attack potential.

168 The strength of a function is rated ‘medium’ if the analysis shows that the function provides adequate protection against attackers possessing a moderate attack potential.

169 The strength of a function is rated ‘high’ if the analysis shows that the function provides adequate protection against attackers possessing a high attack potential.

170 The attack potential is derived from the attacker’s expertise, opportunities, resources, and motivation.

AVA_SOF.1 Strength of TOE security function evaluation

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

Developer action elements:

AVA_SOF.1.1D **The developer shall identify all TOE security mechanisms for which a strength of TOE security function analysis is appropriate.**

AVA_SOF.1.2D **The developer shall perform a strength of TOE security function analysis for each identified mechanism.**

Content and presentation of evidence elements:

AVA_SOF.1.1C **The strength of TOE security function analysis shall determine the impact of the identified TOE security mechanisms on the ability of the TOE security functions to counter the threats.**

AVA_SOF.1.2C **The strength of TOE security function analysis shall demonstrate that the identified strength of the security functions is consistent with the security objectives of the TOE.**

AVA_SOF.1.3C **Each strength claim shall be either basic, medium, or high.**

Evaluator action elements:

AVA_SOF.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AVA_SOF.1.2E **The evaluator shall confirm that all TOE security mechanisms requiring a strength analysis have been identified.**

AVA_SOF.1.3E **The evaluator shall confirm that the strength claims are correct.**

AVA_VLA Vulnerability analysis

Objectives

171 Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or e.g., by flaw hypotheses, could allow malicious users to violate the TSP.

172 Vulnerability analysis deals with the threats that a malicious user will be able to discover flaws that will allow access to resources (e.g., data), allow the ability to interfere with or alter the TSF, or interfere with the authorised capabilities of other users.

Application notes

173 The vulnerability analysis should consider the contents of all the TOE deliverables for the targeted evaluation assurance level.

174 Obvious vulnerabilities are those that allow common attacks or those that might be suggested by the TOE interface description. Obvious vulnerabilities are those in the public domain, details of which should be known to a developer or available from an evaluation oversight body.

175 The evidence identifies all the TOE documentation upon which the search for flaws was based.

AVA_VLA.1 Developer vulnerability analysis

Objectives

176 A vulnerability analysis is performed by the developer to ascertain the presence of "obvious" security vulnerabilities.

177 The objective is to confirm that no identified security vulnerabilities can be exploited in the intended environment for the TOE.

Application notes

178 Obvious vulnerabilities are those which are open to exploitation which requires a minimum of understanding of the TOE, skill, technical sophistication, and resources.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.1.1D **The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.**

AVA_VLA.1.2D **The developer shall document the disposition of identified vulnerabilities.**

Content and presentation of evidence elements:

AVA_VLA.1.1C **The evidence shall show, for each vulnerability, that the vulnerability cannot be exploited in the intended environment for the TOE.**

Evaluator action elements:

AVA_VLA.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AVA_VLA.1.2E **The evaluator shall conduct penetration testing, based on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.**

EAL 3
Methodically tested and checked

ACM Configuration management

179 Configuration management (CM) is an aspect of establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE. CM systems are put in place to ensure the integrity of the configuration items that they control, by providing a method of tracking these configuration items, and by ensuring that only authorised users are capable of changing them.

ACM_CAP CM capabilities

Objectives

180 The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TSF from the early design stages through all subsequent maintenance efforts.

181 The objectives of this family include the following:

a) ensuring that the TSF is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items; and

d) enabling recovery to an earlier version of the TOE, in the event that an error occurs through modification, addition, or deletion of TOE configuration items.

Application notes

182 For ACM_CAP.1 and the higher components, there is a requirement that a configuration list be provided. The configuration list contains all configuration items which are maintained by the CM system.

183 For ACM_CAP.2 and the higher components, there is a requirement that the CM documentation include evidence that the CM system is working properly. An example of such evidence might be audit trail output from the CM system. The

evaluator is responsible for examining such evidence, to determine that it is sufficient to demonstrate proper functionality of the CM system.

184 For ACM_CAP.2 and the higher components, there is a requirement that evidence be provided that all configuration items are being maintained under the CM system. Since a configuration item refers to an item which is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

185 For ACM_CAP.3 and ACM_CAP.4, there is a requirement that the CM system support the generation of all supported versions of the TOE. This provides the ability to recover to a previous known version in the event that an error occurs through modification, addition or deletion of TOE configuration items.

ACM_CAP.2 Authorisation controls

Objectives

186 Clear identification of the TOE is required to determine those items under evaluation that are subject to the criteria requirements.

187 Assurance of TOE integrity may be gained by controlling the ability to modify the TOE configuration items. Ensuring proper functionality and use of the CM system also provides assurance that the CM system is correctly enforcing the integrity of the TOE.

Dependencies:

ACM_SCP.1 Minimal CM coverage

ALC_DVS.1 Identification of security measures

Developer action elements:

ACM_CAP.2.1D The developer shall use a CM system.

ACM_CAP.2.2D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.2.1C **The CM documentation shall include a configuration list and a CM plan.**

ACM_CAP.2.2C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.2.3C The CM documentation shall describe the method used to uniquely identify the TOE configuration items.

ACM_CAP.2.4C **The CM plan shall describe how the CM system is used.**

ACM_CAP.2.5C **The CM documentation shall provide evidence that the CM system is working properly.**

ACM_CAP.2.6C **The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.**

ACM_CAP.2.7C **The CM system shall ensure that only authorised changes are made to the TOE configuration items.**

Evaluator action elements:

ACM_CAP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_SCP CM scope

Objectives

188 The objective is to ensure that all necessary TOE configuration items are tracked by the CM system. This helps to ensure that the integrity of these configuration items is protected through the capabilities of the CM system.

189 The objectives of this family include the following:

a) ensuring that the TOE implementation representation is tracked;

b) ensuring that all necessary documentation, including problem reports, are tracked during development and operation;

c) ensuring that configuration options (e.g. compiler switches) are tracked; and

d) ensuring that development tools are tracked.

Application notes

190 For ACM_SCP.1 and the higher components, there is a requirement that the TOE implementation representation be tracked by the CM system. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

191 For ACM_SCP.2 and ACM_SCP.3, there is a requirement that security flaws be tracked by the CM system. This requires that information regarding previous security flaws and their resolution be maintained, as well as details regarding current security flaws.

192 For ACM_SCP.3, there is a requirement that development tools and other related information be tracked by the CM system. Examples of development tools are programming languages and compilers. Information pertaining to TOE generation

items (such as compiler options, installation/generation options, and build options) is an example of information relating to development tools.

ACM_SCP.1 Minimal CM coverage

Objectives

193 A CM system can control changes only to those items that have been placed under CM. At a minimum, the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation should be placed under CM.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_SCP.1.1D **The developer shall provide CM documentation.**

Content and presentation of evidence elements:

ACM_SCP.1.1C **As a minimum, the following shall be tracked by the CM system: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, and CM documentation.**

ACM_SCP.1.2C **The CM documentation shall describe how configuration items are tracked by the CM system.**

Evaluator action elements:

ACM_SCP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADO Delivery and operation

194 Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

ADO_IGS Installation, generation, and start-up

Objectives

195 Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started in a secure manner as intended by the developer.

Application notes

196 The generation requirements are applicable only to TOEs that provide the ability to generate an operational TOE from source or object code.

197 The installation, generation, and start-up procedures may exist as a separate document, but would typically be grouped with other administrative guidance.

ADO_IGS.1 Installation, generation, and start-up procedures

Dependencies:

AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.1.1D The developer shall document procedures to be used for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.1.1C The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

Evaluator action elements:

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV Development

198 The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. The other family in the development class describes requirements for the internal structure of the TSF.

199 The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, since some of the representations are not necessary for low assurance evaluations.

ADV_FSP Functional specification

Objectives

200 The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is a refinement of the statement of IT functional requirements in the ST of the TOE. The functional specification has to show that all the functional requirements defined in the ST are addressed, and that the TSP is enforced by the TSF.

Application notes

201 In addition to the content indicated in the following requirements, the functional specification shall also include any additional specific detail specified by the documentation notes in the related functional components.

202 The developer must provide evidence that the TSF is completely represented by the functional specification. While a functional specification for the entire TOE would allow an evaluator to determine the TSF boundary, it is not necessary to require that specification when other evidence could be provided to demonstrate the TSF boundary.

203 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the functional specification. In the course of the functional specification evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

204 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

205 While a TSP may represent any policies, TSP models have traditionally represented only subsets of those policies. As a result, the TSP model cannot be treated like every other TSF representation inasmuch as the correspondence between the TSP model to the adjacent abstractions (i.e., TSP and functional specification) may not be complete. As a result, there must be a demonstration of correspondence from the functional specification to the TSP directly, rather than through the intervening representation (i.e., TSP model) where correspondence may be lost. For these reasons, all of the requirements for correspondence between the TSP, TSP model, and functional specification have been included in this family and the

correspondence requirements in the Representation correspondence (ADV_RCR) family do not apply to the TSP and TSP model.

206 Beginning with ADV_FSP.1, requirements are defined to ensure that the functional specification is consistent with the TSP. Beginning with ADV_FSP.2, because there is no requirement for a TSP model in ADV_FSP.1, requirements are defined to describe the rules and characteristics of applicable policies of the TSP in the TSP model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g., state transition, non-interference). For example, rules may be represented as “properties” (e.g., simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects”, and “objects”.

207 Since not all policies can be modeled, given the current state of the art, the requirement indicating which policies shall be modeled is subjective. The PP/ST author should identify specific functions and associated policies that are required to be modeled. At the very least, access control policies are expected to be modeled since they are currently within the state of the art.

ADV_FSP.1 TOE and security policy

Dependencies:

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.1.1D The developer shall provide a functional specification.

ADV_FSP.1.2D The developer shall provide a TSP.

Content and presentation of evidence elements:

ADV_FSP.1.1C The functional specification shall describe the TSF using an informal style.

ADV_FSP.1.2C The functional specification shall include an informal presentation of syntax and semantics of all external TSF interfaces.

ADV_FSP.1.3C The functional specification shall include evidence that demonstrates that the TSF is completely represented.

Evaluator action elements:

ADV_FSP.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2E The evaluator shall determine that the functional specification is consistent with the TSP.

ADV_FSP.1.3E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_HLD High-level design

Objectives

208 The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e., subsystems) and relates these units to the functions that they contain. The high-level design provides assurance that the TOE provides an architecture appropriate to implement the claimed functional requirements.

209 The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function and identifies the security functions enforced by the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

Application notes

210 In addition to the content indicated in the following requirements, the high-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

211 The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

212 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the high-level design. In the course of the high-level design evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

213 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

214 The term “security functionality” is used to represent operations that a subsystem performs that have some effect on the security functions implemented by the TOE. This distinction is made because design constructs, such as subsystems and modules, do not necessarily relate to specific security functions. While a given subsystem may correspond directly to a security function, or even multiple security functions, it is also possible that many subsystems must be combined to implement a single security function.

215 The term “TSP enforcing subsystems” refers to a subsystem that contributes to the enforcement of the TSP.

ADV_HLD.2 Security enforcing high-level design

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_HLD.2.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.2.1C The presentation of the high-level design shall be informal.

ADV_HLD.2.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.2.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.2.4C The high-level design shall identify the interfaces of the subsystems of the TSF.

ADV_HLD.2.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.2.6C **The high-level design shall describe the separation of the TSF into TSP enforcing and other subsystems.**

Evaluator action elements:

ADV_HLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.2.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_RCR Representation correspondence

Objectives

216 The correspondence between the various representations (i.e. functional requirements expressed in the ST, functional specification, high-level design, low-level design, implementation) addresses the correct and complete instantiation of the requirements to the least abstract representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

Application notes

217 The developer must demonstrate to the evaluator that the most detailed, or least abstract, representation of the TSF is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

218 The evaluator must analyse each demonstration of correspondence between abstractions, as well as the results of the analysis of each TSF representation, and then make a determination as to whether the functional requirements in the ST have been satisfied.

219 This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure 5.4, it is intended to address correspondence between the requirements in the ST as well as the TOE summary specification, functional specification, high-level design, low-level design, and implementation representation.

ADV_RCR.1 Informal correspondence demonstration

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.1.1D The developer shall provide evidence that the least abstract TSF representation provided is an accurate, consistent, and complete instantiation of the functional requirements expressed in the ST.

Content and presentation of evidence elements:

ADV_RCR.1.1C For each adjacent pair of TSF representations, the evidence shall demonstrate that all parts of the more abstract representation are refined in the less abstract representation.

ADV_RCR.1.2C For each adjacent pair of TSF representations, the demonstration of correspondence between the representations may be informal.

Evaluator action elements:

ADV_RCR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.1.2E The evaluator shall analyse the correspondence between the functional requirements expressed in the ST and the least abstract representation provided to ensure accuracy, consistency, and completeness.

AGD Guidance documents

220 The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure installation and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

AGD_ADM Administrator guidance

Objectives

221 Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

Application notes

222 The requirements AGD_ADM.1.2C and AGD_ADM.1.11C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

223 The PP/ST author should review the functional components of the PP/ST for guidance on administrator documentation. Those application notes that are relevant

to administrator guidance for understanding and proper application of the security functions should be considered for inclusion in the administrator guidance requirements. An example of an administrator guidance document is a reference manual.

AGD_ADM.1 Administrator guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.2C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.3C The administrator guidance shall contain guidelines on the consistent and effective use of the security functions within the TSF.

AGD_ADM.1.4C The administrator guidance shall describe the difference between two types of functions: those which allow an administrator to control security parameters, and those which allow the administrator to obtain information only.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the administrator's control.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall contain guidelines on how the security functions interact.

AGD_ADM.1.8C The administrator guidance shall contain instructions regarding how to configure the TOE.

AGD_ADM.1.9C The administrator guidance shall describe all configuration options that may be used during secure installation of the TOE.

AGD_ADM.1.10C The administrator guidance shall describe details, sufficient for use, of procedures relevant to the administration of security.

AGD_ADM.1.11C The administrator guidance shall be consistent with all other documents supplied for evaluation.

Evaluator action elements:

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_ADM.1.2E The evaluator shall confirm that the installation procedures result in a secure configuration.

AGD_USR User guidance

Objectives

224 User guidance refers to written material that is intended to be used by nonadministrative (human) users of the TOE. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

225 The user guidance provides a basis for assumptions about the use of the TOE and a measure of confidence that non-malicious users and application providers will understand the secure operation of the TOE and will use it as intended.

Application notes

226 The requirement AGD_USR.1.3.C and AGD_USR.1.5C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the user guidance.

227 The PP/ST author should review the functional components of the PP/ST for guidance on user documentation. Those application notes that are relevant to user guidance aimed at the understanding and proper use of the security functions should be considered for inclusion in the user guidance requirements. Examples of user guidance are reference manuals, user guides, and on-line help.

AGD_USR.1 User guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the TSF and interfaces available to the user.

AGD_USR.1.2C The user guidance shall contain guidelines on the use of security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall describe the interaction between user-visible security functions.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation delivered for evaluation.

Evaluator action elements:

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC Life cycle support

228 Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

ALC_DVS Development security

Objectives

229 Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

Application notes

230 The evaluator should decide whether there is a need for visiting the user's site in order to confirm that the requirements of this family are met.

ALC_DVS.1 Identification of security measures

Dependencies:

No dependencies.

Developer action elements:

ALC_DVS.1.1D **The developer shall produce development security documentation.**

Content and presentation of evidence elements:

ALC_DVS.1.1C **The development security documentation shall describe the physical, procedural, personnel, and other security measures that are used to protect the confidentiality and integrity of the TOE during its development.**

ALC_DVS.1.2C **The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.**

Evaluator action elements:

ALC_DVS.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ALC_DVS.1.2E **The evaluator shall check whether the security measures are being applied.**

ATE Tests

231 The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g., functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of the PP/ ST. Testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. Testing may also be directed toward the internals of the TSF, such as the testing of subsystems and modules against their specifications.

232 The aspects of coverage and depth have been separated from functional tests for reasons of increased flexibility in applying the components of the families. However, the requirements in these three families are intended to be applied together.

233 The independent testing has dependencies on the other families to provide the necessary information to support the requirements, but is primarily concerned with independent evaluator actions.

234 This class does not address penetration testing, which is directed toward finding vulnerabilities that enable a user to violate the security policy. Penetration testing is addressed separately as an aspect of vulnerability assessment in the class AVA.

ATE_COV Coverage

Objectives

235 This family addresses those aspects of testing that deal with completeness of testing. That is, it addresses the extent to which the TOE security functions are tested, whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified, and whether or not the order in which testing proceeds correctly accounts for functional dependencies between the portions of the TOE being tested.

Application notes

236 The specific documentation required by the coverage components will be determined, in most cases, by the documentation stipulated in the level of ATE_FUN that is specified. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_COV.2 Complete coverage - rigorous

Objectives

237 The objective is that testing completely address the security functions.

238 In this component, the objective is to ensure that there is a detailed correspondence between the tests and the security functions.

Application notes

239 The analysis of the test coverage in support of the detailed correspondence can be informal.

Dependencies:

ADV_FSP.1 TOE and security policy

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.2.1D The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements:

ATE_COV.2.1C The analysis of the test coverage shall demonstrate that the tests identified in the test documentation cover the TSF.

ATE_COV.2.2C **The analysis of the test coverage shall demonstrate the correspondence between the security functions and the tests identified in the test documentation.**

Evaluator action elements:

ATE_COV.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT Depth

Objectives

240 The components in this family deal with the level of detail to which the TOE is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.

241 The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internals of the TOE, are more likely to discover any malicious code that has been inserted.

Application notes

242 The specific amount and type of documentation and evidence will, in general, be determined by that required by level of ATE_FUN selected. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_DPT.2 Testing - high level design

Objectives

243 The functional specification of a TOE provides a high level description of the external workings of the TSF. Testing at the level of the functional specification, in order to demonstrate the presence of any flaws, provides assurance that the TSF functional specification has been correctly realised.

244 The subsystems of a TOE provide a high level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

Application notes

245 The functional specification representation is used to express the notion of the most abstract representation of the TSF.

246 The developer is expected to describe the testing of the high level design of the TSF in terms of “subsystems”. The term “subsystem” is used to express the notion of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar notion of decomposition.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.2.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.2.1C **The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TOE operates in accordance with the functional specification, and high level design of the TSF.**

Evaluator action elements:

ATE_DPT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_FUN Functional tests

Objectives

247 Functional testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family “Functional tests” is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through testing.

248 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

Application notes

249 Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures should also show how the test results is derived from the test inputs.

250 The developer shall eliminate all security relevant flaws discovered during testing.

251 The developer shall test the TSF to determine that no new security relevant flaws have been introduced as a result of eliminating discovered security relevant flaws.

ATE_FUN.1 Functional testing

Objectives

252 The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

Dependencies:

ATE_COV.1 Complete coverage - informal

ATE_DPT.1 Testing - functional specification

Developer action elements:

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, and test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function.

ATE_FUN.1.4C The test results in the test documentation shall show the expected results of each test.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each security function operates as specified.

Evaluator action elements:

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND Independent testing

Objectives

253 The objective is to demonstrate that the security functions perform as specified.

254 Additionally, an objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer which results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

Application notes

255 The testing specified in this family can be performed by a party other than the evaluator (e.g., an independent laboratory, an objective consumer organisation).

256 This family deals with the degree to which there is independent functional testing of the TOE. Independent functional testing may take the form of repeating the developer's functional tests, in whole or in part. It may also take the form of the augmentation of the developer's functional tests, either to extend the scope or the depth of the developer's tests.

ATE_IND.2 Independent testing - sample

Objectives

257 The objective is to demonstrate that the security functions perform as specified.

258 In this component, the objective is to select and repeat a sample of the developer testing.

Application notes

259 The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required to run tests. The need for documentation is supported by the dependencies to other assurance families.

260 Additionally, suitability of the TOE for testing may be based on other considerations e.g., the version of the TOE submitted by the developer is not the final version.

261 The developer is required to perform testing and to provide test documentation and test results. This is addressed by the ATE_FUN family.

262 Testing may be selective and shall be based upon all available documentation.

Dependencies:

ADV_FSP.1 TOE and security policy

AGD_USR.1 User guidance

AGD_ADM.1 Administrator guidance

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.2.1C The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test the TSF to confirm that the TSF operates as specified.

ATE_IND.2.3E **The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.**

AVA Vulnerability assessment

263 The class “Vulnerability assessment” encompasses four families: covert channel analysis (AVA_CCA), misuse (AVA_MSU), strength of TOE security functions (AVA_SOF) and vulnerability analysis (AVA_VLA). The class addresses the existence of exploitable covert channels, the misuse or incorrect configuration of the TOE, the ability for all critical security mechanisms to withstand direct attack and the definition and assessment of penetration tests to exploit vulnerabilities introduced in the development or the operation of the TOE.

AVA_MSU Misuse

Objectives

264 Misuse investigates whether the TOE can be configured or used in a manner which is insecure but which an administrator or end-user of the TOE would reasonably believe to be secure.

265 The objective is to minimise the risk of human or other errors in operation which may deactivate, disable, or fail to activate security functions.

266 The objective is to minimise the probability of configuring or installing the TOE in a way which is insecure, without the end user or administrator being able to recognise it.

Application notes

267 Conflicting, misleading or incomplete guidance may result in a user of the TOE believing that the TOE is secure, when it is not. Conflicting guidance can result in vulnerabilities.

268 An example of conflicting guidance would be two guidance instructions which imply different outcomes when the same input is supplied.

269 An example of misleading guidance would be the description of a single guidance instruction which could be parsed in more than one way, one of which may result in an insecure state.

270 An example of completeness would be referencing assertions of dependencies on external security measures e.g., such as external procedural, physical and personnel controls.

AVA_MSU.1 Misuse analysis - obvious flaws

Objectives

271 The objective is to ensure that conflicting guidance in the guidance documentation have been addressed.

Dependencies:

ADO_IGS.1 Installation, generation, and start-up procedures

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.1.1D **The developer shall document an analysis of the guidance documentation for conflicting and incomplete guidance.**

AVA_MSU.1.2D **The developer shall ensure that the guidance documentation contains no misleading or unreasonable guidance.**

Content and presentation of evidence elements:

AVA_MSU.1.1C **The analysis documentation shall provide a rationale that demonstrates that the guidance is not conflicting and is complete.**

Evaluator action elements:

AVA_MSU.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AVA_MSU.1.2E **The evaluator shall determine that there is no misleading or unreasonable guidance in the guidance documentation.**

AVA_MSU.1.3E **The evaluator shall repeat any procedures in the guidance documentation to ensure that they produce the documented results.**

AVA_SOF Strength of TOE security functions

Objectives

272 Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security functions claim.

Application notes

273 Security functions are implemented by security mechanisms. For example, a password mechanism can be used in the implementation of the identification and authentication security function.

274 The strength of TOE security functions evaluation is performed at the level of the security mechanism, but its results provide knowledge about the ability of the related security function to counter the identified threats.

275 The strength of a function is rated ‘basic’ if the analysis shows that the function provides adequate protection against unintended or casual breach of TOE security by attackers possessing a low attack potential.

276 The strength of a function is rated ‘medium’ if the analysis shows that the function provides adequate protection against attackers possessing a moderate attack potential.

277 The strength of a function is rated ‘high’ if the analysis shows that the function provides adequate protection against attackers possessing a high attack potential.

278 The attack potential is derived from the attacker’s expertise, opportunities, resources, and motivation.

AVA_SOF.1 Strength of TOE security function evaluation

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

Developer action elements:

AVA_SOF.1.1D The developer shall identify all TOE security mechanisms for which a strength of TOE security function analysis is appropriate.

AVA_SOF.1.2D The developer shall perform a strength of TOE security function analysis for each identified mechanism.

Content and presentation of evidence elements:

AVA_SOF.1.1C The strength of TOE security function analysis shall determine the impact of the identified TOE security mechanisms on the ability of the TOE security functions to counter the threats.

AVA_SOF.1.2C The strength of TOE security function analysis shall demonstrate that the identified strength of the security functions is consistent with the security objectives of the TOE.

AVA_SOF.1.3C Each strength claim shall be either basic, medium, or high.

Evaluator action elements:

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that all TOE security mechanisms requiring a strength analysis have been identified.

AVA_SOF.1.3E The evaluator shall confirm that the strength claims are correct.

AVA_VLA Vulnerability analysis

Objectives

279 Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or e.g., by flaw hypotheses, could allow malicious users to violate the TSP.

280 Vulnerability analysis deals with the threats that a malicious user will be able to discover flaws that will allow access to resources (e.g., data), allow the ability to interfere with or alter the TSF, or interfere with the authorised capabilities of other users.

Application notes

281 The vulnerability analysis should consider the contents of all the TOE deliverables for the targeted evaluation assurance level.

282 Obvious vulnerabilities are those that allow common attacks or those that might be suggested by the TOE interface description. Obvious vulnerabilities are those in the public domain, details of which should be known to a developer or available from an evaluation oversight body.

283 The evidence identifies all the TOE documentation upon which the search for flaws was based.

AVA_VLA.1 Developer vulnerability analysis

Objectives

284 A vulnerability analysis is performed by the developer to ascertain the presence of "obvious" security vulnerabilities.

285 The objective is to confirm that no identified security vulnerabilities can be exploited in the intended environment for the TOE.

Application notes

286 Obvious vulnerabilities are those which are open to exploitation which requires a minimum of understanding of the TOE, skill, technical sophistication, and resources.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.1.1D The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.

AVA_VLA.1.2D The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.1.1C The evidence shall show, for each vulnerability, that the vulnerability cannot be exploited in the intended environment for the TOE.

Evaluator action elements:

AVA_VLA.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.1.2E The evaluator shall conduct penetration testing, based on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

EAL 4
Methodically designed, tested, and reviewed

ACM Configuration management

287 Configuration management (CM) is an aspect of establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE. CM systems are put in place to ensure the integrity of the configuration items that they control, by providing a method of tracking these configuration items, and by ensuring that only authorised users are capable of changing them.

ACM_AUT CM automation

Objectives

288 The objective of introducing automated CM tools is to increase the efficiency of the CM system, by simultaneously increasing the reliability of the CM system and reducing the cost of operating it. While both automated and manual CM systems can be bypassed, ignored, or insufficient to prevent unauthorised modification, automated systems are less susceptible to human error or negligence. In addition, while a manual CM system can accomplish all of the same things that an automated system can, manual systems are typically more costly to operate on an ongoing basis.

Application notes

289 For ACM_AUT.1 and ACM_AUT.2, there is a requirement that the automated CM system control changes to the implementation representation of the TOE. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

ACM_AUT.1 Partial CM automation

Objectives

290 In development environments where the implementation representation is complex or is being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are performed by authorised developers before their application.

It is the objective of this component to ensure that the implementation representation is controlled through automated means.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_AUT.1.1D **The developer shall provide a CM plan.**

Content and presentation of evidence elements:

ACM_AUT.1.1C **The CM plan shall describe the automated tools used in the CM system.**

ACM_AUT.1.2C **The CM plan shall describe how the automated tools are used in the CM system.**

ACM_AUT.1.3C **The CM system shall provide an automated means to ensure that only authorised changes are made to the TOE implementation representation.**

ACM_AUT.1.4C **The CM system shall provide an automated means to support the generation of any supported TSF from its implementation representation.**

ACM_AUT.1.5C **The CM system shall provide an automated means to support the comparison of any two supported TSF versions, to ascertain the changes.**

Evaluator action elements:

ACM_AUT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ACM_CAP CM capabilities

Objectives

291 The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TSF from the early design stages through all subsequent maintenance efforts.

292 The objectives of this family include the following:

a) ensuring that the TSF is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items; and

d) enabling recovery to an earlier version of the TOE, in the event that an error occurs through modification, addition, or deletion of TOE configuration items.

Application notes

293 For ACM_CAP.1 and the higher components, there is a requirement that a configuration list be provided. The configuration list contains all configuration items which are maintained by the CM system.

294 For ACM_CAP.2 and the higher components, there is a requirement that the CM documentation include evidence that the CM system is working properly. An example of such evidence might be audit trail output from the CM system. The evaluator is responsible for examining such evidence, to determine that it is sufficient to demonstrate proper functionality of the CM system.

295 For ACM_CAP.2 and the higher components, there is a requirement that evidence be provided that all configuration items are being maintained under the CM system. Since a configuration item refers to an item which is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

296 For ACM_CAP.3 and ACM_CAP.4, there is a requirement that the CM system support the generation of all supported versions of the TOE. This provides the ability to recover to a previous known version in the event that an error occurs through modification, addition or deletion of TOE configuration items.

ACM_CAP.3 Generation support and acceptance procedures

Objectives

297 Clear identification of the TOE is required to determine those items under evaluation that are subject to the criteria requirements.

298 Assurance of TOE integrity may be gained by controlling the ability to modify the TOE configuration items. Ensuring proper functionality and use of the CM system also provides assurance that the CM system is correctly enforcing the integrity of the TOE.

299 The ability to generate previous but still supported versions of the TOE is necessary for the resolution of any new flaws discovered during operation.

300 The purpose of acceptance procedures is to confirm that any creation or modification of TSF configuration items is authorised.

Dependencies:

ACM_SCP.1 Minimal CM coverage

ALC_DVS.1 Identification of security measures

Developer action elements:

ACM_CAP.3.1D The developer shall use a CM system.

ACM_CAP.3.2D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.3.1C **The CM documentation shall include a configuration list, a CM plan, and an acceptance plan.**

ACM_CAP.3.2C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.3.3C The CM documentation shall describe the method used to uniquely identify the TOE configuration items.

ACM_CAP.3.4C The CM plan shall describe how the CM system is used.

ACM_CAP.3.5C The CM documentation shall provide evidence that the CM system is working properly.

ACM_CAP.3.6C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.3.7C The CM system shall ensure that only authorised changes are made to the TOE configuration items.

ACM_CAP.3.8C **The CM system shall support the generation of all supported versions of the TOE.**

ACM_CAP.3.9C **The acceptance plan shall describe the procedures used to accept modified or newly created TSF configuration items as part of the TOE.**

Evaluator action elements:

ACM_CAP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_SCP CM scope

Objectives

301 The objective is to ensure that all necessary TOE configuration items are tracked by the CM system. This helps to ensure that the integrity of these configuration items is protected through the capabilities of the CM system.

302 The objectives of this family include the following:

a) ensuring that the TOE implementation representation is tracked;

b) ensuring that all necessary documentation, including problem reports, are tracked during development and operation;

c) ensuring that configuration options (e.g. compiler switches) are tracked; and

d) ensuring that development tools are tracked.

Application notes

303 For ACM_SCP.1 and the higher components, there is a requirement that the TOE implementation representation be tracked by the CM system. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

304 For ACM_SCP.2 and ACM_SCP.3, there is a requirement that security flaws be tracked by the CM system. This requires that information regarding previous security flaws and their resolution be maintained, as well as details regarding current security flaws.

305 For ACM_SCP.3, there is a requirement that development tools and other related information be tracked by the CM system. Examples of development tools are programming languages and compilers. Information pertaining to TOE generation items (such as compiler options, installation/generation options, and build options) is an example of information relating to development tools.

ACM_SCP.2 Problem tracking CM coverage

Objectives

306 A CM system can control changes only to those items that have been placed under CM. At a minimum, the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation should be placed under CM.

307 The ability to track security flaws under CM ensures that security flaw reports are not lost or forgotten, and allows a developer to track security flaws to their resolution.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_SCP.2.1D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_SCP.2.1C **As a minimum, the following shall be tracked by the CM system: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation, and security flaws.**

ACM_SCP.2.2C The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator action elements:

ACM_SCP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO Delivery and operation

308 Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

ADO_IGS Installation, generation, and start-up

Objectives

309 Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started in a secure manner as intended by the developer.

Application notes

310 The generation requirements are applicable only to TOEs that provide the ability to generate an operational TOE from source or object code.

311 The installation, generation, and start-up procedures may exist as a separate document, but would typically be grouped with other administrative guidance.

ADO_IGS.1 Installation, generation, and start-up procedures

Dependencies:

AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.1.1D The developer shall document procedures to be used for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.1.1C The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

Evaluator action elements:

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV Development

312 The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. The other family in the development class describes requirements for the internal structure of the TSF.

313 The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, since some of the representations are not necessary for low assurance evaluations.

ADV_FSP Functional specification

Objectives

314 The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is a refinement of the statement of IT functional requirements in the ST of the TOE. The functional specification has to show that all

the functional requirements defined in the ST are addressed, and that the TSP is enforced by the TSF.

Application notes

315 In addition to the content indicated in the following requirements, the functional specification shall also include any additional specific detail specified by the documentation notes in the related functional components.

316 The developer must provide evidence that the TSF is completely represented by the functional specification. While a functional specification for the entire TOE would allow an evaluator to determine the TSF boundary, it is not necessary to require that specification when other evidence could be provided to demonstrate the TSF boundary.

317 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the functional specification. In the course of the functional specification evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

318 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

319 While a TSP may represent any policies, TSP models have traditionally represented only subsets of those policies. As a result, the TSP model cannot be treated like every other TSF representation inasmuch as the correspondence between the TSP model to the adjacent abstractions (i.e., TSP and functional specification) may not be complete. As a result, there must be a demonstration of correspondence from the functional specification to the TSP directly, rather than through the intervening representation (i.e., TSP model) where correspondence may be lost. For these reasons, all of the requirements for correspondence between the TSP, TSP model, and functional specification have been included in this family and the correspondence requirements in the Representation correspondence (ADV_RCR) family do not apply to the TSP and TSP model.

320 Beginning with ADV_FSP.1, requirements are defined to ensure that the functional specification is consistent with the TSP. Beginning with ADV_FSP.2, because there is no requirement for a TSP model in ADV_FSP.1, requirements are defined to describe the rules and characteristics of applicable policies of the TSP in the TSP

model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g., state transition, non-interference). For example, rules may be represented as “properties” (e.g., simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects”, and “objects”.

321 Since not all policies can be modeled, given the current state of the art, the requirement indicating which policies shall be modeled is subjective. The PP/ST author should identify specific functions and associated policies that are required to be modeled. At the very least, access control policies are expected to be modeled since they are currently within the state of the art.

ADV_FSP.2 Informal security policy model

Dependencies:

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.2.1D The developer shall provide a functional specification.

ADV_FSP.2.2D The developer shall provide a TSP.

ADV_FSP.2.3D **The developer shall provide an informal TSP model.**

ADV_FSP.2.4D **The developer shall provide a demonstration of correspondence between the informal TSP model and the functional specification.**

Content and presentation of evidence elements:

ADV_FSP.2.1C The functional specification shall describe the TSF using an informal style.

ADV_FSP.2.2C The functional specification shall include an informal presentation of syntax and semantics of all external TSF interfaces.

ADV_FSP.2.3C The functional specification shall include evidence that demonstrates that the TSF is completely represented.

ADV_FSP.2.4C **The demonstration of correspondence between the informal TSP model and the functional specification shall describe how the functional specification satisfies the informal TSP model.**

ADV_FSP.2.5C **The demonstration of correspondence between the informal TSP model and the functional specification shall show that there are no security functions in the functional specification that conflict with the informal TSP model.**

ADV_FSP.2.6C **The informal TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.**

ADV_FSP.2.7C **The informal TSP model shall include a rationale that demonstrates that policies of the TSP that are modeled are satisfied by the informal TSP model.**

ADV_FSP.2.8C **The informal TSP model shall justify that all policies of the TSP that can be modeled are represented in the informal TSP model.**

Evaluator action elements:

ADV_FSP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.2.2E The evaluator shall determine that the functional specification is consistent with the TSP.

ADV_FSP.2.3E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_HLD High-level design

Objectives

322 The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e., subsystems) and relates these units to the functions that they contain. The high-level design provides assurance that the TOE provides an architecture appropriate to implement the claimed functional requirements.

323 The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function and identifies the security functions enforced by the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

Application notes

324 In addition to the content indicated in the following requirements, the high-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

325 The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

326 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the high-level design. In the course of the high-level design evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

327 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

328 The term “security functionality” is used to represent operations that a subsystem performs that have some effect on the security functions implemented by the TOE. This distinction is made because design constructs, such as subsystems and modules, do not necessarily relate to specific security functions. While a given subsystem may correspond directly to a security function, or even multiple security functions, it is also possible that many subsystems must be combined to implement a single security function.

329 The term “TSP enforcing subsystems” refers to a subsystem that contributes to the enforcement of the TSP.

ADV_HLD.2 Security enforcing high-level design

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_HLD.2.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.2.1C The presentation of the high-level design shall be informal.

ADV_HLD.2.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.2.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.2.4C The high-level design shall identify the interfaces of the subsystems of the TSF.

ADV_HLD.2.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.2.6C The high-level design shall describe the separation of the TSF into TSP enforcing and other subsystems.

Evaluator action elements:

ADV_HLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.2.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_IMP Implementation representation

Objectives

330 The description of the implementation in the form of source code, firmware, hardware drawings, etc. captures the detailed internal workings of the TSF in support of analysis.

Application notes

331 The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. Source code which is then compiled or a hardware drawing which is used to build the actual hardware are examples of parts of an implementation representation.

332 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the implementation. In the course of the implementation evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a more abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis is necessary. However, since the implementation is the least abstract representation it is likely that further analysis cannot be performed, unless the TSF representations have not been evaluated in a usual order (i.e., most abstract to least abstract). If requirements are not addressed after the analysis of all TSF representations, this represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

333 In all cases, it is important that the evaluator evaluates the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

334 It is expected that evaluators will use the implementation to directly support other evaluation activities (e.g., vulnerability analysis, test coverage analysis). It is expected that PP/ST authors will select a component that requires that the implementation is complete and comprehensible enough to address the needs of all other requirements included in the PP/ST.

ADV_IMP.1 Subset of the implementation of the TSF

Application notes

335 The PP/ST author should identify the subset of the implementation representation to be delivered. If a specific subset of the source code/hardware drawing to be delivered has not been specified by the PP/ST author, the evaluator has the option of requesting a subset of the source code/hardware drawings for analysis.

336 The intent is not an open ended invitation for the evaluator to demand implementation representations, but rather that the evaluator may request implementation representations that may support the demonstration that functional requirements have been met. For example, see the application notes for this family of assurance components.

Dependencies:

ADV_LLD.1 Descriptive low-level design

ADV_RCR.1 Informal correspondence demonstration

ALC_TAT.1 Well defined development tools

Developer action elements:

ADV_IMP.1.1D **The developer shall provide the implementation representations for a selected subset of the TSF.**

Content and presentation of evidence elements:

ADV_IMP.1.1C **The implementation representations shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.**

Evaluator action elements:

ADV_IMP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_IMP.1.2E **The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.**

ADV_LLD Low-level design

Objectives

337 The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined.

338 For each module of the TSF, the low-level design describes its purpose, function, interfaces, dependencies, and the implementation of any TSP enforcing functions.

Application notes

339 In addition to the content indicated in the following requirements, the low-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

340 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the low-level design. In the course of the low-level design evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

341 In all cases, it is important that the evaluator evaluates the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

342 The term “TSP enforcing function” refers to any function that contributes to TSP enforcement. The term “TSP enforcing modules” similarly refers to any module that contributes to TSP enforcement.

ADV_LLD.1 Descriptive low-level design

Dependencies:

ADV_HLD.1 Descriptive high-level design

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_LLD.1.1D **The developer shall provide the low-level design of the TSF.**

Content and presentation of evidence elements:

ADV_LLD.1.1C **The presentation of the low-level design shall be informal.**

ADV_LLD.1.2C **The low-level design shall describe the TSF in terms of modules.**

ADV_LLD.1.3C **The low-level design shall describe the purpose of each module.**

ADV_LLD.1.4C **The low-level design shall define the interrelationships between the modules in terms of provided functionality and dependencies on other modules.**

ADV_LLD.1.5C **The low-level design shall describe the implementation of all TSP enforcing functions.**

ADV_LLD.1.6C **The low-level design shall describe the interfaces of each module in terms of their syntax and semantics.**

ADV_LLD.1.7C **The low-level design shall provide a demonstration that the TSF is completely represented.**

ADV_LLD.1.8C **The low-level design shall identify the interfaces of the modules of the TSF visible at the external interface of the TSF.**

Evaluator action elements:

ADV_LLD.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_LLD.1.2E **The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.**

ADV_RCR Representation correspondence

Objectives

343 The correspondence between the various representations (i.e. functional requirements expressed in the ST, functional specification, high-level design, low-level design, implementation) addresses the correct and complete instantiation of the requirements to the least abstract representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

Application notes

344 The developer must demonstrate to the evaluator that the most detailed, or least abstract, representation of the TSF is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

345 The evaluator must analyse each demonstration of correspondence between abstractions, as well as the results of the analysis of each TSF representation, and then make a determination as to whether the functional requirements in the ST have been satisfied.

346 This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure 5.4, it is intended to address correspondence between the requirements in the ST as well as the TOE summary specification, functional specification, high-level design, low-level design, and implementation representation.

ADV_RCR.1 Informal correspondence demonstration

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.1.1D The developer shall provide evidence that the least abstract TSF representation provided is an accurate, consistent, and complete instantiation of the functional requirements expressed in the ST.

Content and presentation of evidence elements:

ADV_RCR.1.1C For each adjacent pair of TSF representations, the evidence shall demonstrate that all parts of the more abstract representation are refined in the less abstract representation.

ADV_RCR.1.2C For each adjacent pair of TSF representations, the demonstration of correspondence between the representations may be informal.

Evaluator action elements:

ADV_RCR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.1.2E The evaluator shall analyse the correspondence between the functional requirements expressed in the ST and the least abstract representation provided to ensure accuracy, consistency, and completeness.

AGD Guidance documents

347 The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure installation and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

AGD_ADM Administrator guidance

Objectives

348 Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

Application notes

349 The requirements AGD_ADM.1.2C and AGD_ADM.1.11C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

350 The PP/ST author should review the functional components of the PP/ST for guidance on administrator documentation. Those application notes that are relevant to administrator guidance for understanding and proper application of the security functions should be considered for inclusion in the administrator guidance requirements. An example of an administrator guidance document is a reference manual.

AGD_ADM.1 Administrator guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.2C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.3C The administrator guidance shall contain guidelines on the consistent and effective use of the security functions within the TSF.

AGD_ADM.1.4C The administrator guidance shall describe the difference between two types of functions: those which allow an administrator to control security parameters, and those which allow the administrator to obtain information only.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the administrator's control.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall contain guidelines on how the security functions interact.

AGD_ADM.1.8C The administrator guidance shall contain instructions regarding how to configure the TOE.

AGD_ADM.1.9C The administrator guidance shall describe all configuration options that may be used during secure installation of the TOE.

AGD_ADM.1.10C The administrator guidance shall describe details, sufficient for use, of procedures relevant to the administration of security.

AGD_ADM.1.11C The administrator guidance shall be consistent with all other documents supplied for evaluation.

Evaluator action elements:

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_ADM.1.2E The evaluator shall confirm that the installation procedures result in a secure configuration.

AGD_USR User guidance

Objectives

351 User guidance refers to written material that is intended to be used by nonadministrative (human) users of the TOE. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

352 The user guidance provides a basis for assumptions about the use of the TOE and a measure of confidence that non-malicious users and application providers will understand the secure operation of the TOE and will use it as intended.

Application notes

353 The requirement AGD_USR.1.3.C and AGD_USR.1.5C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the user guidance.

354 The PP/ST author should review the functional components of the PP/ST for guidance on user documentation. Those application notes that are relevant to user guidance aimed at the understanding and proper use of the security functions should be considered for inclusion in the user guidance requirements. Examples of user guidance are reference manuals, user guides, and on-line help.

AGD_USR.1 User guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the TSF and interfaces available to the user.

AGD_USR.1.2C The user guidance shall contain guidelines on the use of security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall describe the interaction between user-visible security functions.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation delivered for evaluation.

Evaluator action elements:

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC Life cycle support

355 Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

ALC_DVS Development security

Objectives

356 Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

Application notes

357 The evaluator should decide whether there is a need for visiting the user's site in order to confirm that the requirements of this family are met.

ALC_DVS.1 Identification of security measures

Dependencies:

No dependencies.

Developer action elements:

ALC_DVS.1.1D The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.1.1C The development security documentation shall describe the physical, procedural, personnel, and other security measures that are used to protect the confidentiality and integrity of the TOE during its development.

ALC_DVS.1.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

Evaluator action elements:

ALC_DVS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.1.2E The evaluator shall check whether the security measures are being applied.

ALC_LCD Life cycle definition

Objectives

358 Poorly controlled development and maintenance can result in a flawed implementation of a TOE (or a TOE that does not meet all of its security requirements). This, in turn, results in security violations. Therefore, it is important that a model for the development and maintenance of a TOE be established as early as possible in the TOE's life-cycle.

359 Using a model for the development and maintenance of a TOE does not guarantee that the TOE will be free of flaws, nor does it guarantee that the TOE will meet all of its security functional requirements. It is possible that the model chosen was insufficient or inadequate and therefore no benefits in the quality of the TOE could be observed. Using a life-cycle model that has been approved by some group of experts (e.g., academic experts, standards bodies) improves the chances that the development and maintenance models will contribute to the overall quality of the TOE.

Application notes

360 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming relevant after the completion of the evaluation, its evaluation adds assurance through an analysis the life-cycle information for the TOE provided at the time of the evaluation.

361 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE.

362 A standardised life-cycle model is a model that has been approved by some group of experts (e.g., academic experts, standards bodies).

363 A measurable life-cycle model is a model with some arithmetic parameters so that e.g. the coding standards can be measured.

ALC_LCD.1 Developer defined life-cycle model

Dependencies:

No dependencies.

Developer action elements:

ALC_LCD.1.1D **The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.**

ALC_LCD.1.2D **The developer shall produce life-cycle definition documentation.**

Content and presentation of evidence elements:

ALC_LCD.1.1C **The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.**

Evaluator action elements:

ALC_LCD.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ALC_TAT Tools and techniques

Objectives

364 Tools and techniques is an aspect of selecting tools which are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to programming languages, documentation, implementation standards, and other parts of the TOE like supporting runtime libraries.

Application notes

365 There is a requirement for well-defined development tools. These are tools which have been shown to be well understood and applicable without the need for intensive further clarification. For example, programming languages and computer aided design (CAD) systems that are based on an a standard published by standards bodies are considered to be well-defined.

366 Tools and techniques distinguishes between the implementation standards applied by the developer and the implementation standards for “all parts of the TOE” which additionally includes third party software, hardware, or firmware.

367 The requirement in ALC_TAT.1.2C is specifically applicable to programming languages so as to ensure that all statements in the source code have an unambiguous meaning.

ALC_TAT.1 Well defined development tools

Dependencies:

No dependencies.

Developer action elements:

ALC_TAT.1.1D **The developer shall identify the development tools being used for the TOE.**

ALC_TAT.1.2D **The developer shall document the selected implementation dependent options of the development tools.**

Content and presentation of evidence elements:

ALC_TAT.1.1C **Any development tools used for implementation shall be well-defined.**

ALC_TAT.1.2C **The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.**

Evaluator action elements:

ALC_TAT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE Tests

368 The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g., functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of the PP/ ST. Testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. Testing may also be directed toward the internals of the TSF, such as the testing of subsystems and modules against their specifications.

369 The aspects of coverage and depth have been separated from functional tests for reasons of increased flexibility in applying the components of the families. However, the requirements in these three families are intended to be applied together.

370 The independent testing has dependencies on the other families to provide the necessary information to support the requirements, but is primarily concerned with independent evaluator actions.

371 This class does not address penetration testing, which is directed toward finding vulnerabilities that enable a user to violate the security policy. Penetration testing is addressed separately as an aspect of vulnerability assessment in the class AVA.

ATE_COV Coverage

Objectives

372 This family addresses those aspects of testing that deal with completeness of testing. That is, it addresses the extent to which the TOE security functions are tested, whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified, and whether or not the order in which testing proceeds correctly accounts for functional dependencies between the portions of the TOE being tested.

Application notes

373 The specific documentation required by the coverage components will be determined, in most cases, by the documentation stipulated in the level of ATE_FUN that is specified. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_COV.2 Complete coverage - rigorous

Objectives

374 The objective is that testing completely address the security functions.

375 In this component, the objective is to ensure that there is a detailed correspondence between the tests and the security functions.

Application notes

376 The analysis of the test coverage in support of the detailed correspondence can be informal.

Dependencies:

ADV_FSP.1 TOE and security policy

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.2.1D The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements:

ATE_COV.2.1C The analysis of the test coverage shall demonstrate that the tests identified in the test documentation cover the TSF.

ATE_COV.2.2C The analysis of the test coverage shall demonstrate the correspondence between the security functions and the tests identified in the test documentation.

Evaluator action elements:

ATE_COV.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT Depth

Objectives

377 The components in this family deal with the level of detail to which the TOE is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.

378 The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internals of the TOE, are more likely to discover any malicious code that has been inserted.

Application notes

379 The specific amount and type of documentation and evidence will, in general, be determined by that required by level of ATE_FUN selected. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_DPT.2 Testing - high level design

Objectives

380 The functional specification of a TOE provides a high level description of the external workings of the TSF. Testing at the level of the functional specification, in order to demonstrate the presence of any flaws, provides assurance that the TSF functional specification has been correctly realised.

381 The subsystems of a TOE provide a high level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

Application notes

382 The functional specification representation is used to express the notion of the most abstract representation of the TSF.

383 The developer is expected to describe the testing of the high level design of the TSF in terms of “subsystems”. The term “subsystem” is used to express the notion of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar notion of decomposition.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.2.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.2.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TOE operates in accordance with the functional specification, and high level design of the TSF.

Evaluator action elements:

ATE_DPT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_FUN Functional tests

Objectives

384 Functional testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family “Functional tests” is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through testing.

385 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

Application notes

386 Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures should also show how the test results is derived from the test inputs.

387 The developer shall eliminate all security relevant flaws discovered during testing.

388 The developer shall test the TSF to determine that no new security relevant flaws have been introduced as a result of eliminating discovered security relevant flaws.

ATE_FUN.1 Functional testing

Objectives

389 The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

Dependencies:

ATE_COV.1 Complete coverage - informal

ATE_DPT.1 Testing - functional specification

Developer action elements:

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, and test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function.

ATE_FUN.1.4C The test results in the test documentation shall show the expected results of each test.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each security function operates as specified.

Evaluator action elements:

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND Independent testing

Objectives

390 The objective is to demonstrate that the security functions perform as specified.

391 Additionally, an objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer which results in the incorrect implementation

of the specifications, or overlooks code that is non-compliant with the specifications.

Application notes

392 The testing specified in this family can be performed by a party other than the evaluator (e.g., an independent laboratory, an objective consumer organisation).

393 This family deals with the degree to which there is independent functional testing of the TOE. Independent functional testing may take the form of repeating the developer's functional tests, in whole or in part. It may also take the form of the augmentation of the developer's functional tests, either to extend the scope or the depth of the developer's tests.

ATE_IND.2 Independent testing - sample

Objectives

394 The objective is to demonstrate that the security functions perform as specified.

395 In this component, the objective is to select and repeat a sample of the developer testing.

Application notes

396 The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required to run tests. The need for documentation is supported by the dependencies to other assurance families.

397 Additionally, suitability of the TOE for testing may be based on other considerations e.g., the version of the TOE submitted by the developer is not the final version.

398 The developer is required to perform testing and to provide test documentation and test results. This is addressed by the ATE_FUN family.

399 Testing may be selective and shall be based upon all available documentation.

Dependencies:

ADV_FSP.1 TOE and security policy

AGD_USR.1 User guidance

AGD_ADM.1 Administrator guidance

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.2.1C The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test the TSF to confirm that the TSF operates as specified.

ATE_IND.2.3E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

AVA Vulnerability assessment

400 The class “Vulnerability assessment” encompasses four families: covert channel analysis (AVA_CCA), misuse (AVA_MSU), strength of TOE security functions (AVA_SOF) and vulnerability analysis (AVA_VLA). The class addresses the existence of exploitable covert channels, the misuse or incorrect configuration of the TOE, the ability for all critical security mechanisms to withstand direct attack and the definition and assessment of penetration tests to exploit vulnerabilities introduced in the development or the operation of the TOE.

AVA_MSU Misuse

Objectives

401 Misuse investigates whether the TOE can be configured or used in a manner which is insecure but which an administrator or end-user of the TOE would reasonably believe to be secure.

402 The objective is to minimise the risk of human or other errors in operation which may deactivate, disable, or fail to activate security functions.

403 The objective is to minimise the probability of configuring or installing the TOE in a way which is insecure, without the end user or administrator being able to recognise it.

Application notes

404 Conflicting, misleading or incomplete guidance may result in a user of the TOE believing that the TOE is secure, when it is not. Conflicting guidance can result in vulnerabilities.

405 An example of conflicting guidance would be two guidance instructions which imply different outcomes when the same input is supplied.

406 An example of misleading guidance would be the description of a single guidance instruction which could be parsed in more than one way, one of which may result in an insecure state.

407 An example of completeness would be referencing assertions of dependencies on external security measures e.g., such as external procedural, physical and personnel controls.

AVA_MSU.2 Misuse analysis - independent verification

Objectives

408 The objective is to ensure that conflicting guidance in the guidance documentation have been addressed.

409 In this component, the objective is to provide additional assurance by performing an independent analysis.

Dependencies:

ADO_IGS.1 Installation, generation, and start-up procedures

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.2.1D The developer shall document an analysis of the guidance documentation for conflicting and incomplete guidance.

AVA_MSU.2.2D The developer shall ensure that the guidance documentation contains no misleading or unreasonable guidance.

Content and presentation of evidence elements:

AVA_MSU.2.1C The analysis documentation shall provide a rationale that demonstrates that the guidance is not conflicting and is complete.

Evaluator action elements:

AVA_MSU.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E The evaluator shall determine that there is no misleading or unreasonable guidance in the guidance documentation.

AVA_MSU.2.3E The evaluator shall repeat any procedures in the guidance documentation to ensure that they produce the documented results.

AVA_MSU.2.4E **The evaluator shall perform independent testing to confirm that the TOE can be configured and operated securely using only the guidance documentation.**

AVA_SOF Strength of TOE security functions

Objectives

410 Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security functions claim.

Application notes

411 Security functions are implemented by security mechanisms. For example, a password mechanism can be used in the implementation of the identification and authentication security function.

412 The strength of TOE security functions evaluation is performed at the level of the security mechanism, but its results provide knowledge about the ability of the related security function to counter the identified threats.

413 The strength of a function is rated ‘basic’ if the analysis shows that the function provides adequate protection against unintended or casual breach of TOE security by attackers possessing a low attack potential.

414 The strength of a function is rated ‘medium’ if the analysis shows that the function provides adequate protection against attackers possessing a moderate attack potential.

415 The strength of a function is rated ‘high’ if the analysis shows that the function provides adequate protection against attackers possessing a high attack potential.

416 The attack potential is derived from the attacker’s expertise, opportunities, resources, and motivation.

AVA_SOF.1 Strength of TOE security function evaluation

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

Developer action elements:

AVA_SOF.1.1D The developer shall identify all TOE security mechanisms for which a strength of TOE security function analysis is appropriate.

AVA_SOF.1.2D The developer shall perform a strength of TOE security function analysis for each identified mechanism.

Content and presentation of evidence elements:

AVA_SOF.1.1C The strength of TOE security function analysis shall determine the impact of the identified TOE security mechanisms on the ability of the TOE security functions to counter the threats.

AVA_SOF.1.2C The strength of TOE security function analysis shall demonstrate that the identified strength of the security functions is consistent with the security objectives of the TOE.

AVA_SOF.1.3C Each strength claim shall be either basic, medium, or high.

Evaluator action elements:

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that all TOE security mechanisms requiring a strength analysis have been identified.

AVA_SOF.1.3E The evaluator shall confirm that the strength claims are correct.

AVA_VLA Vulnerability analysis

Objectives

417 Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or e.g., by flaw hypotheses, could allow malicious users to violate the TSP.

418 Vulnerability analysis deals with the threats that a malicious user will be able to discover flaws that will allow access to resources (e.g., data), allow the ability to interfere with or alter the TSF, or interfere with the authorised capabilities of other users.

Application notes

419 The vulnerability analysis should consider the contents of all the TOE deliverables for the targeted evaluation assurance level.

420 Obvious vulnerabilities are those that allow common attacks or those that might be suggested by the TOE interface description. Obvious vulnerabilities are those in the public domain, details of which should be known to a developer or available from an evaluation oversight body.

421 The evidence identifies all the TOE documentation upon which the search for flaws was based.

AVA_VLA.2 Independent vulnerability analysis

Objectives

422 A vulnerability analysis is performed by the developer to ascertain the presence of "obvious" security vulnerabilities.

423 The objective is to confirm that no identified security vulnerabilities can be exploited in the intended environment for the TOE.

424 An independent vulnerability analysis is performed by the evaluator, which goes beyond the "obvious" security vulnerabilities. The analysis considers the deliverables available for the targeted evaluation assurance level.

Application notes

425 Obvious vulnerabilities are those which are open to exploitation which requires a minimum of understanding of the TOE, skill, technical sophistication, and resources.

426 Independent vulnerability analysis is based on fairly detailed technical information. The attacker is assumed to be only reasonably familiar with the specific implementation of the TOE. The attacker is presumed to have a reasonable level of technical sophistication.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.2.1D The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.

AVA_VLA.2.2D The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.2.1C The evidence shall show, for each vulnerability, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.2.2C **The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.**

Evaluator action elements:

AVA_VLA.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.2.2E The evaluator shall conduct penetration testing, based on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

AVA_VLA.2.3E **The evaluator shall perform an independent vulnerability analysis.**

AVA_VLA.2.4E **The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of identified vulnerabilities in the target environment.**

AVA_VLA.2.5E **The evaluator shall determine that the TOE is resistant to obvious penetration attacks.**

EAL 5
Semiformally designed and tested

ACM Configuration management

427 Configuration management (CM) is an aspect of establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE. CM systems are put in place to ensure the integrity of the configuration items that they control, by providing a method of tracking these configuration items, and by ensuring that only authorised users are capable of changing them.

ACM_AUT CM automation

Objectives

428 The objective of introducing automated CM tools is to increase the efficiency of the CM system, by simultaneously increasing the reliability of the CM system and reducing the cost of operating it. While both automated and manual CM systems can be bypassed, ignored, or insufficient to prevent unauthorised modification, automated systems are less susceptible to human error or negligence. In addition, while a manual CM system can accomplish all of the same things that an automated system can, manual systems are typically more costly to operate on an ongoing basis.

Application notes

429 For ACM_AUT.1 and ACM_AUT.2, there is a requirement that the automated CM system control changes to the implementation representation of the TOE. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

ACM_AUT.1 Partial CM automation

Objectives

430 In development environments where the implementation representation is complex or is being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are performed by authorised developers before their application.

It is the objective of this component to ensure that the implementation representation is controlled through automated means.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_AUT.1.1D The developer shall provide a CM plan.

Content and presentation of evidence elements:

ACM_AUT.1.1C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.1.2C The CM plan shall describe how the automated tools are used in the CM system.

ACM_AUT.1.3C The CM system shall provide an automated means to ensure that only authorised changes are made to the TOE implementation representation.

ACM_AUT.1.4C The CM system shall provide an automated means to support the generation of any supported TSF from its implementation representation.

ACM_AUT.1.5C The CM system shall provide an automated means to support the comparison of any two supported TSF versions, to ascertain the changes.

Evaluator action elements:

ACM_AUT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP CM capabilities

Objectives

431 The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TSF from the early design stages through all subsequent maintenance efforts.

432 The objectives of this family include the following:

a) ensuring that the TSF is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items; and

d) enabling recovery to an earlier version of the TOE, in the event that an error occurs through modification, addition, or deletion of TOE configuration items.

Application notes

433 For ACM_CAP.1 and the higher components, there is a requirement that a configuration list be provided. The configuration list contains all configuration items which are maintained by the CM system.

434 For ACM_CAP.2 and the higher components, there is a requirement that the CM documentation include evidence that the CM system is working properly. An example of such evidence might be audit trail output from the CM system. The evaluator is responsible for examining such evidence, to determine that it is sufficient to demonstrate proper functionality of the CM system.

435 For ACM_CAP.2 and the higher components, there is a requirement that evidence be provided that all configuration items are being maintained under the CM system. Since a configuration item refers to an item which is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

436 For ACM_CAP.3 and ACM_CAP.4, there is a requirement that the CM system support the generation of all supported versions of the TOE. This provides the ability to recover to a previous known version in the event that an error occurs through modification, addition or deletion of TOE configuration items.

ACM_CAP.3 Generation support and acceptance procedures

Objectives

437 Clear identification of the TOE is required to determine those items under evaluation that are subject to the criteria requirements.

438 Assurance of TOE integrity may be gained by controlling the ability to modify the TOE configuration items. Ensuring proper functionality and use of the CM system also provides assurance that the CM system is correctly enforcing the integrity of the TOE.

439 The ability to generate previous but still supported versions of the TOE is necessary for the resolution of any new flaws discovered during operation.

440 The purpose of acceptance procedures is to confirm that any creation or modification of TSF configuration items is authorised.

Dependencies:

ACM_SCP.1 Minimal CM coverage

ALC_DVS.1 Identification of security measures

Developer action elements:

ACM_CAP.3.1D The developer shall use a CM system.

ACM_CAP.3.2D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.3.1C The CM documentation shall include a configuration list, a CM plan, and an acceptance plan.

ACM_CAP.3.2C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.3.3C The CM documentation shall describe the method used to uniquely identify the TOE configuration items.

ACM_CAP.3.4C The CM plan shall describe how the CM system is used.

ACM_CAP.3.5C The CM documentation shall provide evidence that the CM system is working properly.

ACM_CAP.3.6C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.3.7C The CM system shall ensure that only authorised changes are made to the TOE configuration items.

ACM_CAP.3.8C The CM system shall support the generation of all supported versions of the TOE.

ACM_CAP.3.9C The acceptance plan shall describe the procedures used to accept modified or newly created TSF configuration items as part of the TOE.

Evaluator action elements:

ACM_CAP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_SCP CM scope

Objectives

441 The objective is to ensure that all necessary TOE configuration items are tracked by the CM system. This helps to ensure that the integrity of these configuration items is protected through the capabilities of the CM system.

442 The objectives of this family include the following:

a) ensuring that the TOE implementation representation is tracked;

b) ensuring that all necessary documentation, including problem reports, are tracked during development and operation;

c) ensuring that configuration options (e.g. compiler switches) are tracked; and

d) ensuring that development tools are tracked.

Application notes

443 For ACM_SCP.1 and the higher components, there is a requirement that the TOE implementation representation be tracked by the CM system. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

444 For ACM_SCP.2 and ACM_SCP.3, there is a requirement that security flaws be tracked by the CM system. This requires that information regarding previous security flaws and their resolution be maintained, as well as details regarding current security flaws.

445 For ACM_SCP.3, there is a requirement that development tools and other related information be tracked by the CM system. Examples of development tools are programming languages and compilers. Information pertaining to TOE generation items (such as compiler options, installation/generation options, and build options) is an example of information relating to development tools.

ACM_SCP.3 Development tools CM coverage

Objectives

446 A CM system can control changes only to those items that have been placed under CM. At a minimum, the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation should be placed under CM.

447 The ability to track security flaws under CM ensures that security flaw reports are not lost or forgotten, and allows a developer to track security flaws to their resolution.

448 Development tools play an important role in ensuring the production of a quality version of the TSF. Therefore, it is important to control modifications to these tools.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_SCP.3.1D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_SCP.3.1C **As a minimum, the following shall be tracked by the CM system: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation, security flaws, and development tools and related information.**

ACM_SCP.3.2C The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator action elements:

ACM_SCP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO Delivery and operation

449 Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

ADO_IGS Installation, generation, and start-up

Objectives

450 Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started in a secure manner as intended by the developer.

Application notes

451 The generation requirements are applicable only to TOEs that provide the ability to generate an operational TOE from source or object code.

452 The installation, generation, and start-up procedures may exist as a separate document, but would typically be grouped with other administrative guidance.

ADO_IGS.1 Installation, generation, and start-up procedures

Dependencies:

AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.1.1D The developer shall document procedures to be used for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.1.1C The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

Evaluator action elements:

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV Development

453 The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. The other family in the development class describes requirements for the internal structure of the TSF.

454 The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, since some of the representations are not necessary for low assurance evaluations.

ADV_FSP Functional specification

Objectives

455 The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is a refinement of the statement of IT functional requirements in the ST of the TOE. The functional specification has to show that all the functional requirements defined in the ST are addressed, and that the TSP is enforced by the TSF.

Application notes

456 In addition to the content indicated in the following requirements, the functional specification shall also include any additional specific detail specified by the documentation notes in the related functional components.

457 The developer must provide evidence that the TSF is completely represented by the functional specification. While a functional specification for the entire TOE would

allow an evaluator to determine the TSF boundary, it is not necessary to require that specification when other evidence could be provided to demonstrate the TSF boundary.

458 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the functional specification. In the course of the functional specification evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

459 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

460 While a TSP may represent any policies, TSP models have traditionally represented only subsets of those policies. As a result, the TSP model cannot be treated like every other TSF representation inasmuch as the correspondence between the TSP model to the adjacent abstractions (i.e., TSP and functional specification) may not be complete. As a result, there must be a demonstration of correspondence from the functional specification to the TSP directly, rather than through the intervening representation (i.e., TSP model) where correspondence may be lost. For these reasons, all of the requirements for correspondence between the TSP, TSP model, and functional specification have been included in this family and the correspondence requirements in the Representation correspondence (ADV_RCR) family do not apply to the TSP and TSP model.

461 Beginning with ADV_FSP.1, requirements are defined to ensure that the functional specification is consistent with the TSP. Beginning with ADV_FSP.2, because there is no requirement for a TSP model in ADV_FSP.1, requirements are defined to describe the rules and characteristics of applicable policies of the TSP in the TSP model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g., state transition, non-interference). For example, rules may be represented as “properties” (e.g., simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects”, and “objects”.

462 Since not all policies can be modeled, given the current state of the art, the requirement indicating which policies shall be modeled is subjective. The PP/ST author should identify specific functions and associated policies that are required to

be modeled. At the very least, access control policies are expected to be modeled since they are currently within the state of the art.

ADV_FSP.4 Formal security policy model

Application notes

463 The requirement for both an informal and semiformal functional specification is necessary to allow an evaluator to effectively comprehend and evaluate the semiformal representation using the informal representation for support.

Dependencies:

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.4.1D The developer shall provide a functional specification.

ADV_FSP.4.2D The developer shall provide a TSP.

ADV_FSP.4.3D **The developer shall provide a formal TSP model.**

ADV_FSP.4.4D **The developer shall provide a demonstration of correspondence between the formal TSP model and the functional specification.**

Content and presentation of evidence elements:

ADV_FSP.4.1C **The functional specification shall describe the TSF using both an informal and semiformal style.**

ADV_FSP.4.2C **The functional specification shall include both an informal and semiformal presentation of syntax, effects, exceptions, error messages, and semantics of all external TSF interfaces.**

ADV_FSP.4.3C The functional specification shall include evidence that demonstrates that the TSF is completely represented.

ADV_FSP.4.4C **The demonstration of correspondence between the formal TSP model and the functional specification shall describe how the functional specification satisfies the formal TSP model.**

ADV_FSP.4.5C **The demonstration of correspondence between the formal TSP model and the functional specification shall show that there are no security functions in the functional specification that conflict with the formal TSP model.**

ADV_FSP.4.6C **The formal TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.**

ADV_FSP.4.7C **The formal TSP model shall include a rationale that demonstrates that policies of the TSP that are modeled are satisfied by the formal TSP model.**

ADV_FSP.4.8C **The formal TSP model shall justify that all policies of the TSP that can be modeled are represented in the formal TSP model.**

Evaluator action elements:

ADV_FSP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.4.2E The evaluator shall determine that the functional specification is consistent with the TSP.

ADV_FSP.4.3E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_HLD High-level design

Objectives

464 The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e., subsystems) and relates these units to the functions that they contain. The high-level design provides assurance that the TOE provides an architecture appropriate to implement the claimed functional requirements.

465 The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function and identifies the security functions enforced by the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

Application notes

466 In addition to the content indicated in the following requirements, the high-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

467 The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

468 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the high-level design. In the course of the high-level design evaluation there are essentially three types of evaluator

determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

469 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

470 The term “security functionality” is used to represent operations that a subsystem performs that have some effect on the security functions implemented by the TOE. This distinction is made because design constructs, such as subsystems and modules, do not necessarily relate to specific security functions. While a given subsystem may correspond directly to a security function, or even multiple security functions, it is also possible that many subsystems must be combined to implement a single security function.

471 The term “TSP enforcing subsystems” refers to a subsystem that contributes to the enforcement of the TSP.

ADV_HLD.3 Semiformal high-level design

Dependencies:

ADV_FSP.3 Semiformal security policy model

ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements:

ADV_HLD.3.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.3.1C **The presentation of the high-level design shall be semiformal.**

ADV_HLD.3.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.3.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.3.4C The high-level design shall identify the interfaces of the subsystems of the TSF.

ADV_HLD.3.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.3.6C The high-level design shall describe the separation of the TSF into TSP enforcing and other subsystems.

Evaluator action elements:

ADV_HLD.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.3.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_IMP Implementation representation

Objectives

472 The description of the implementation in the form of source code, firmware, hardware drawings, etc. captures the detailed internal workings of the TSF in support of analysis.

Application notes

473 The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. Source code which is then compiled or a hardware drawing which is used to build the actual hardware are examples of parts of an implementation representation.

474 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the implementation. In the course of the implementation evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a more abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis is necessary. However, since the implementation is the least abstract representation it is likely that further analysis cannot be performed, unless the TSF representations have not been evaluated in a usual order (i.e., most abstract to least abstract). If requirements are not addressed after the analysis of all TSF representations, this represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

475 In all cases, it is important that the evaluator evaluates the TSF as a unit since in many cases the security functions must cooperate to meet specific functional

requirements and also each security function must not interfere with the operation of any other security function.

476 It is expected that evaluators will use the implementation to directly support other evaluation activities (e.g., vulnerability analysis, test coverage analysis). It is expected that PP/ST authors will select a component that requires that the implementation is complete and comprehensible enough to address the needs of all other requirements included in the PP/ST.

ADV_IMP.2 Implementation of the TSF

Dependencies:

ADV_LLD.1 Descriptive low-level design

ADV_RCR.1 Informal correspondence demonstration

ALC_TAT.2 Compliance with implementation standards

Developer action elements:

ADV_IMP.2.1D **The developer shall provide the implementation representations for the entire TSF.**

Content and presentation of evidence elements:

ADV_IMP.2.1C The implementation representations shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.2.2C **The implementation representations shall describe the relationships between all portions of the implementation.**

Evaluator action elements:

ADV_IMP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.2.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_INT TSF internals

Objectives

477 This family of components deals with the internal structure of the TSF. Requirements are established for modularity, the layering of the software architecture to separate levels of abstraction and minimisation of circular dependencies, and the minimisation from the TSF of software that is not TSP enforcing.

478 Modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Thus, a modular design provides the basis for determining the scope of interaction with other elements of the TSF, provides for increased assurance that unexpected effects do not occur, and also provides the basis for designing and evaluating test suites.

479 Design complexity affects how difficult it is to understand the design of the TOE. The simpler the design, the more assurance is gained that there are no hidden vulnerabilities in the design and that the high-level protection requirements are accurately and completely instantiated in the lower level design and the implementation.

480 Design complexity minimisation provides a part of the assurance that the code is understood; the less complex the code in the TSF, the greater the likelihood that the design of the TSF is comprehensible. Design complexity minimisation is a key characteristic of a reference validation mechanism.

Application notes

481 The term “relevant representation” is used in these components to cover the need for an evaluator to check for the appropriate issue (e.g., modularity, complexity) at whichever level of representation (e.g., high-level design, implementation) the requirements are being invoked.

482 The term “portions of the TSF” is used to represent parts of the TSF with a varying granularity based on the available TSF representations. The functional specification allows identification in terms of interfaces, the high-level design allows identification in terms of subsystems, the low-level design allows identification in terms of modules, and the implementation representation allows identification in terms of implementation units (e.g., source code files).

ADV_INT.1 Modularity

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

Developer action elements:

ADV_INT.1.1D **The developer shall design the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.**

ADV_INT.1.2D **The developer shall provide an architectural description.**

Content and presentation of evidence elements:

ADV_INT.1.1C **The architectural description shall identify the modules of the TSF.**

ADV_INT.1.2C **The architectural description shall describe the purpose, interface, parameters, and effects of each module in the TSF.**

ADV_INT.1.3C **The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.**

Evaluator action elements:

ADV_INT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_INT.1.2E **The evaluator shall check the relevant representations for compliance with the architectural description.**

ADV_LLD Low-level design

Objectives

483 The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined.

484 For each module of the TSF, the low-level design describes its purpose, function, interfaces, dependencies, and the implementation of any TSP enforcing functions.

Application notes

485 In addition to the content indicated in the following requirements, the low-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

486 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the low-level design. In the course of the low-level design evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

487 In all cases, it is important that the evaluator evaluates the TSF as a unit since in many cases the security functions must cooperate to meet specific functional

requirements and also each security function must not interfere with the operation of any other security function.

488 The term “TSP enforcing function” refers to any function that contributes to TSP enforcement. The term “TSP enforcing modules” similarly refers to any module that contributes to TSP enforcement.

ADV_LLD.1 Descriptive low-level design

Dependencies:

ADV_HLD.1 Descriptive high-level design

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_LLD.1.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements:

ADV_LLD.1.1C The presentation of the low-level design shall be informal.

ADV_LLD.1.2C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.1.3C The low-level design shall describe the purpose of each module.

ADV_LLD.1.4C The low-level design shall define the interrelationships between the modules in terms of provided functionality and dependencies on other modules.

ADV_LLD.1.5C The low-level design shall describe the implementation of all TSP enforcing functions.

ADV_LLD.1.6C The low-level design shall describe the interfaces of each module in terms of their syntax and semantics.

ADV_LLD.1.7C The low-level design shall provide a demonstration that the TSF is completely represented.

ADV_LLD.1.8C The low-level design shall identify the interfaces of the modules of the TSF visible at the external interface of the TSF.

Evaluator action elements:

ADV_LLD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.1.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_RCR Representation correspondence

Objectives

489 The correspondence between the various representations (i.e. functional requirements expressed in the ST, functional specification, high-level design, low-level design, implementation) addresses the correct and complete instantiation of the requirements to the least abstract representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

Application notes

490 The developer must demonstrate to the evaluator that the most detailed, or least abstract, representation of the TSF is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

491 The evaluator must analyse each demonstration of correspondence between abstractions, as well as the results of the analysis of each TSF representation, and then make a determination as to whether the functional requirements in the ST have been satisfied.

492 This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure 5.4, it is intended to address correspondence between the requirements in the ST as well as the TOE summary specification, functional specification, high-level design, low-level design, and implementation representation.

ADV_RCR.2 Semiformal correspondence demonstration

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.2.1D The developer shall provide evidence that the least abstract TSF representation provided is an accurate, consistent, and complete instantiation of the functional requirements expressed in the ST.

Content and presentation of evidence elements:

ADV_RCR.2.1C For each adjacent pair of TSF representations, the evidence shall demonstrate that all parts of the more abstract representation are refined in the less abstract representation.

ADV_RCR.2.2C **For each adjacent pair of TSF representations, where portions of both representations are at least semiformally specified, the demonstration of**

correspondence between those portions of the representations shall be semiformal.

ADV_RCR.2.3C **For each adjacent pair of TSF representations, where portions of either representation are informally specified the demonstration of correspondence between those portions of the representations may be informal.**

Evaluator action elements:

ADV_RCR.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.2.2E The evaluator shall analyse the correspondence between the functional requirements expressed in the ST and the least abstract representation provided to ensure accuracy, consistency, and completeness.

AGD Guidance documents

493 The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure installation and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

AGD_ADM Administrator guidance

Objectives

494 Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

Application notes

495 The requirements AGD_ADM.1.2C and AGD_ADM.1.11C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

496 The PP/ST author should review the functional components of the PP/ST for guidance on administrator documentation. Those application notes that are relevant to administrator guidance for understanding and proper application of the security functions should be considered for inclusion in the administrator guidance

requirements. An example of an administrator guidance document is a reference manual.

AGD_ADM.1 Administrator guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.2C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.3C The administrator guidance shall contain guidelines on the consistent and effective use of the security functions within the TSF.

AGD_ADM.1.4C The administrator guidance shall describe the difference between two types of functions: those which allow an administrator to control security parameters, and those which allow the administrator to obtain information only.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the administrator's control.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall contain guidelines on how the security functions interact.

AGD_ADM.1.8C The administrator guidance shall contain instructions regarding how to configure the TOE.

AGD_ADM.1.9C The administrator guidance shall describe all configuration options that may be used during secure installation of the TOE.

AGD_ADM.1.10C The administrator guidance shall describe details, sufficient for use, of procedures relevant to the administration of security.

AGD_ADM.1.11C The administrator guidance shall be consistent with all other documents supplied for evaluation.

Evaluator action elements:

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_ADM.1.2E The evaluator shall confirm that the installation procedures result in a secure configuration.

AGD_USR User guidance

Objectives

497 User guidance refers to written material that is intended to be used by nonadministrative (human) users of the TOE. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

498 The user guidance provides a basis for assumptions about the use of the TOE and a measure of confidence that non-malicious users and application providers will understand the secure operation of the TOE and will use it as intended.

Application notes

499 The requirement AGD_USR.1.3.C and AGD_USR.1.5C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the user guidance.

500 The PP/ST author should review the functional components of the PP/ST for guidance on user documentation. Those application notes that are relevant to user guidance aimed at the understanding and proper use of the security functions should be considered for inclusion in the user guidance requirements. Examples of user guidance are reference manuals, user guides, and on-line help.

AGD_USR.1 User guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the TSF and interfaces available to the user.

AGD_USR.1.2C The user guidance shall contain guidelines on the use of security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall describe the interaction between user-visible security functions.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation delivered for evaluation.

Evaluator action elements:

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC Life cycle support

501 Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

ALC_DVS Development security

Objectives

502 Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

Application notes

503 The evaluator should decide whether there is a need for visiting the user's site in order to confirm that the requirements of this family are met.

ALC_DVS.1 Identification of security measures

Dependencies:

No dependencies.

Developer action elements:

ALC_DVS.1.1D The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.1.1C The development security documentation shall describe the physical, procedural, personnel, and other security measures that are used to protect the confidentiality and integrity of the TOE during its development.

ALC_DVS.1.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

Evaluator action elements:

ALC_DVS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.1.2E The evaluator shall check whether the security measures are being applied.

ALC_LCD Life cycle definition

Objectives

504 Poorly controlled development and maintenance can result in a flawed implementation of a TOE (or a TOE that does not meet all of its security requirements). This, in turn, results in security violations. Therefore, it is important that a model for the development and maintenance of a TOE be established as early as possible in the TOE's life-cycle.

505 Using a model for the development and maintenance of a TOE does not guarantee that the TOE will be free of flaws, nor does it guarantee that the TOE will meet all of its security functional requirements. It is possible that the model chosen was insufficient or inadequate and therefore no benefits in the quality of the TOE could be observed. Using a life-cycle model that has been approved by some group of experts (e.g., academic experts, standards bodies) improves the chances that the development and maintenance models will contribute to the overall quality of the TOE.

Application notes

506 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming relevant after the completion of the evaluation, its evaluation adds assurance through an analysis the life-cycle information for the TOE provided at the time of the evaluation.

507 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE.

508 A standardised life-cycle model is a model that has been approved by some group of experts (e.g., academic experts, standards bodies).

509 A measurable life-cycle model is a model with some arithmetic parameters so that e.g. the coding standards can be measured.

ALC_LCD.2 Standardised life-cycle model

Dependencies:

No dependencies.

Developer action elements:

ALC_LCD.2.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.2.2D The developer shall produce life-cycle definition documentation.

ALC_LCD.2.3D **The developer shall use a standardised life-cycle model to develop and maintain the TOE.**

Content and presentation of evidence elements:

ALC_LCD.2.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.2.2C **The life-cycle definition documentation shall explain why the model was chosen and how it is used to develop and maintain the TOE.**

ALC_LCD.2.3C **The life-cycle definition documentation shall demonstrate compliance with the standardised life-cycle model.**

Evaluator action elements:

ALC_LCD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT Tools and techniques

Objectives

510 Tools and techniques is an aspect of selecting tools which are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to programming languages, documentation, implementation standards, and other parts of the TOE like supporting runtime libraries.

Application notes

511 There is a requirement for well-defined development tools. These are tools which have been shown to be well understood and applicable without the need for intensive further clarification. For example, programming languages and computer aided design (CAD) systems that are based on an a standard published by standards bodies are considered to be well-defined.

512 Tools and techniques distinguishes between the implementation standards applied by the developer and the implementation standards for "all parts of the TOE" which additionally includes third party software, hardware, or firmware.

513 The requirement in ALC_TAT.1.2C is specifically applicable to programming languages so as to ensure that all statements in the source code have an unambiguous meaning.

ALC_TAT.2 Compliance with implementation standards

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements:

ALC_TAT.2.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.2.2D The developer shall document the selected implementation dependent options of the development tools.

ALC_TAT.2.3D **The developer shall describe the implementation standards to be applied.**

Content and presentation of evidence elements:

ALC_TAT.2.1C Any development tools used for implementation shall be well-defined.

ALC_TAT.2.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

Evaluator action elements:

ALC_TAT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.2.2E **The evaluator shall confirm that the implementation standards have been applied.**

ATE Tests

514 The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g., functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of the PP/ ST. Testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. Testing may also be directed toward the internals of the TSF, such as the testing of subsystems and modules against their specifications.

515 The aspects of coverage and depth have been separated from functional tests for reasons of increased flexibility in applying the components of the families. However, the requirements in these three families are intended to be applied together.

516 The independent testing has dependencies on the other families to provide the necessary information to support the requirements, but is primarily concerned with independent evaluator actions.

517 This class does not address penetration testing, which is directed toward finding vulnerabilities that enable a user to violate the security policy. Penetration testing is addressed separately as an aspect of vulnerability assessment in the class AVA.

ATE_COV Coverage

Objectives

518 This family addresses those aspects of testing that deal with completeness of testing. That is, it addresses the extent to which the TOE security functions are tested, whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified, and whether or not the order in which testing proceeds correctly accounts for functional dependencies between the portions of the TOE being tested.

Application notes

519 The specific documentation required by the coverage components will be determined, in most cases, by the documentation stipulated in the level of ATE_FUN that is specified. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_COV.2 Complete coverage - rigorous

Objectives

520 The objective is that testing completely address the security functions.

521 In this component, the objective is to ensure that there is a detailed correspondence between the tests and the security functions.

Application notes

522 The analysis of the test coverage in support of the detailed correspondence can be informal.

Dependencies:

ADV_FSP.1 TOE and security policy

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.2.1D The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements:

ATE_COV.2.1C The analysis of the test coverage shall demonstrate that the tests identified in the test documentation cover the TSF.

ATE_COV.2.2C The analysis of the test coverage shall demonstrate the correspondence between the security functions and the tests identified in the test documentation.

Evaluator action elements:

ATE_COV.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT Depth

Objectives

523 The components in this family deal with the level of detail to which the TOE is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.

524 The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internals of the TOE, are more likely to discover any malicious code that has been inserted.

Application notes

525 The specific amount and type of documentation and evidence will, in general, be determined by that required by level of ATE_FUN selected. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_DPT.3 Testing - low level design

Objectives

526 The functional specification of a TOE provides a high level description of the external workings of the TSF. Testing at the level of the functional specification, in order to demonstrate the presence of any flaws, provides assurance that the TSF functional specification has been correctly realised.

527 The subsystems of a TOE provide a high level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

528 The modules of a TOE provide a description of the internal workings of the TSF. Testing at the level of the modules, in order to demonstrate the presence of any flaws, provides assurance that the TSF modules have been correctly realised.

Application notes

529 The functional specification representation is used to express the notion of the most abstract representation of the TSF.

530 The developer is expected to describe the testing of the high level design of the TSF in terms of “subsystems”. The term “subsystem” is used to express the notion of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar notion of decomposition.

531 The developer is expected to describe the testing of the low level design of the TSF in terms of “modules”. The term “modules” is used to express the notion of decomposing each of the “subsystems” of the TSF into a relatively small number of parts. While the developer is not required to actually have “modules”, the developer is expected to represent a similar notion of decomposition.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ADV_LLD.1 Descriptive low-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.3.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.3.1C **The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TOE operates in accordance with the functional specification, high level design, and low level design of the TSF.**

Evaluator action elements:

ATE_DPT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_FUN Functional tests

Objectives

532 Functional testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family “Functional tests” is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through testing.

533 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

Application notes

534 Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures should also show how the test results is derived from the test inputs.

535 The developer shall eliminate all security relevant flaws discovered during testing.

536 The developer shall test the TSF to determine that no new security relevant flaws have been introduced as a result of eliminating discovered security relevant flaws.

ATE_FUN.1 Functional testing

Objectives

537 The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

Dependencies:

ATE_COV.1 Complete coverage - informal

ATE_DPT.1 Testing - functional specification

Developer action elements:

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, and test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function.

ATE_FUN.1.4C The test results in the test documentation shall show the expected results of each test.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each security function operates as specified.

Evaluator action elements:

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND Independent testing

Objectives

538 The objective is to demonstrate that the security functions perform as specified.

539 Additionally, an objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer which results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

Application notes

540 The testing specified in this family can be performed by a party other than the evaluator (e.g., an independent laboratory, an objective consumer organisation).

541 This family deals with the degree to which there is independent functional testing of the TOE. Independent functional testing may take the form of repeating the developer's functional tests, in whole or in part. It may also take the form of the

augmentation of the developer's functional tests, either to extend the scope or the depth of the developer's tests.

ATE_IND.2 Independent testing - sample

Objectives

542 The objective is to demonstrate that the security functions perform as specified.

543 In this component, the objective is to select and repeat a sample of the developer testing.

Application notes

544 The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required to run tests. The need for documentation is supported by the dependencies to other assurance families.

545 Additionally, suitability of the TOE for testing may be based on other considerations e.g., the version of the TOE submitted by the developer is not the final version.

546 The developer is required to perform testing and to provide test documentation and test results. This is addressed by the ATE_FUN family.

547 Testing may be selective and shall be based upon all available documentation.

Dependencies:

ADV_FSP.1 TOE and security policy

AGD_USR.1 User guidance

AGD_ADM.1 Administrator guidance

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.2.1C The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test the TSF to confirm that the TSF operates as specified.

ATE_IND.2.3E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

AVA Vulnerability assessment

548 The class “Vulnerability assessment” encompasses four families: covert channel analysis (AVA_CCA), misuse (AVA_MSU), strength of TOE security functions (AVA_SOF) and vulnerability analysis (AVA_VLA). The class addresses the existence of exploitable covert channels, the misuse or incorrect configuration of the TOE, the ability for all critical security mechanisms to withstand direct attack and the definition and assessment of penetration tests to exploit vulnerabilities introduced in the development or the operation of the TOE.

AVA_CCA Covert channel analysis

Objectives

549 Covert channel analysis is carried out to determine the existence and potential capacity of unintended signalling channels that may be exploited by malicious code.

550 The assurance requirements address the threat that unintended and exploitable signalling paths exist which may be exercised to violate the security policy.

Application notes

551 Channel capacity estimations are based upon informal engineering measurements, as well as actual test measurements.

552 Details of the assumptions upon which the covert channel analysis is based shall be given, e.g., processor speed, configuration, memory, and cache size.

553 Test parameters details are (e.g., processor speed, memory and cache size), relevant configuration parameters, how the channel was exercised, used to obtain the capacity during testing.

554 The selective validation of the covert channel analysis through testing allows the evaluator the opportunity to verify any aspect of the covert channel analysis (e.g., identification, capacity estimation, elimination, monitoring, and exploitation scenarios). This does not impose a requirement to demonstrate the entire set of covert channel analysis results.

555 If there are no information flow control policies in the ST, this family of assurance requirements is no longer applicable since this family only applies to information flow control policies. Even if there are no specific functional requirements (e.g., FDP_IFF.1 to FDP_IFF.3) for eliminating, limiting, or monitoring covert channels, this family still requires the identification of covert channels.

AVA_CCA.1 Covert channel analysis

Objectives

556 The objective is to identify covert channels which are identifiable through analysis.

557 In this component, the objective is to perform informal search for covert storage channels.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_IMP.1 Subset of the implementation of the TSF

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_CCA.1.1D **The developer shall conduct a search for covert channels for each information flow control policy.**

AVA_CCA.1.2D **The developer shall provide covert channel analysis documentation.**

Content and presentation of evidence elements:

AVA_CCA.1.1C **The analysis documentation shall identify covert channels.**

AVA_CCA.1.2C **The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.**

AVA_CCA.1.3C **The analysis documentation shall describe all assumptions made during the covert channel analysis.**

AVA_CCA.1.4C **The analysis documentation shall describe the method used for estimating channel capacity, which shall be based on worst case scenarios.**

AVA_CCA.1.5C **The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.**

AVA_CCA.1.6C **The analysis documentation shall provide evidence that the method used to identify covert channels is informal.**

Evaluator action elements:

AVA_CCA.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AVA_CCA.1.2E **The evaluator shall confirm that the results of the covert channels analysis meet the functional requirements.**

AVA_CCA.1.3E **The evaluator shall selectively validate the covert channel analysis through testing.**

AVA_MSU Misuse

Objectives

558 Misuse investigates whether the TOE can be configured or used in a manner which is insecure but which an administrator or end-user of the TOE would reasonably believe to be secure.

559 The objective is to minimise the risk of human or other errors in operation which may deactivate, disable, or fail to activate security functions.

560 The objective is to minimise the probability of configuring or installing the TOE in a way which is insecure, without the end user or administrator being able to recognise it.

Application notes

561 Conflicting, misleading or incomplete guidance may result in a user of the TOE believing that the TOE is secure, when it is not. Conflicting guidance can result in vulnerabilities.

562 An example of conflicting guidance would be two guidance instructions which imply different outcomes when the same input is supplied.

563 An example of misleading guidance would be the description of a single guidance instruction which could be parsed in more than one way, one of which may result in an insecure state.

564 An example of completeness would be referencing assertions of dependencies on external security measures e.g., such as external procedural, physical and personnel controls.

AVA_MSU.2 Misuse analysis - independent verification

Objectives

565 The objective is to ensure that conflicting guidance in the guidance documentation have been addressed.

566 In this component, the objective is to provide additional assurance by performing an independent analysis.

Dependencies:

ADO_IGS.1 Installation, generation, and start-up procedures

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.2.1D The developer shall document an analysis of the guidance documentation for conflicting and incomplete guidance.

AVA_MSU.2.2D The developer shall ensure that the guidance documentation contains no misleading or unreasonable guidance.

Content and presentation of evidence elements:

AVA_MSU.2.1C The analysis documentation shall provide a rationale that demonstrates that the guidance is not conflicting and is complete.

Evaluator action elements:

AVA_MSU.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E The evaluator shall determine that there is no misleading or unreasonable guidance in the guidance documentation.

AVA_MSU.2.3E The evaluator shall repeat any procedures in the guidance documentation to ensure that they produce the documented results.

AVA_MSU.2.4E The evaluator shall perform independent testing to confirm that the TOE can be configured and operated securely using only the guidance documentation.

AVA_SOF Strength of TOE security functions

Objectives

567 Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security functions claim.

Application notes

568 Security functions are implemented by security mechanisms. For example, a password mechanism can be used in the implementation of the identification and authentication security function.

569 The strength of TOE security functions evaluation is performed at the level of the security mechanism, but its results provide knowledge about the ability of the related security function to counter the identified threats.

570 The strength of a function is rated ‘basic’ if the analysis shows that the function provides adequate protection against unintended or casual breach of TOE security by attackers possessing a low attack potential.

571 The strength of a function is rated ‘medium’ if the analysis shows that the function provides adequate protection against attackers possessing a moderate attack potential.

572 The strength of a function is rated ‘high’ if the analysis shows that the function provides adequate protection against attackers possessing a high attack potential.

573 The attack potential is derived from the attacker’s expertise, opportunities, resources, and motivation.

AVA_SOF.1 Strength of TOE security function evaluation

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

Developer action elements:

AVA_SOF.1.1D The developer shall identify all TOE security mechanisms for which a strength of TOE security function analysis is appropriate.

AVA_SOF.1.2D The developer shall perform a strength of TOE security function analysis for each identified mechanism.

Content and presentation of evidence elements:

AVA_SOF.1.1C The strength of TOE security function analysis shall determine the impact of the identified TOE security mechanisms on the ability of the TOE security functions to counter the threats.

AVA_SOF.1.2C The strength of TOE security function analysis shall demonstrate that the identified strength of the security functions is consistent with the security objectives of the TOE.

AVA_SOF.1.3C Each strength claim shall be either basic, medium, or high.

Evaluator action elements:

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that all TOE security mechanisms requiring a strength analysis have been identified.

AVA_SOF.1.3E The evaluator shall confirm that the strength claims are correct.

AVA_VLA Vulnerability analysis

Objectives

574 Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or e.g., by flaw hypotheses, could allow malicious users to violate the TSP.

575 Vulnerability analysis deals with the threats that a malicious user will be able to discover flaws that will allow access to resources (e.g., data), allow the ability to interfere with or alter the TSF, or interfere with the authorised capabilities of other users.

Application notes

576 The vulnerability analysis should consider the contents of all the TOE deliverables for the targeted evaluation assurance level.

577 Obvious vulnerabilities are those that allow common attacks or those that might be suggested by the TOE interface description. Obvious vulnerabilities are those in the public domain, details of which should be known to a developer or available from an evaluation oversight body.

578 The evidence identifies all the TOE documentation upon which the search for flaws was based.

AVA_VLA.3 Relatively resistant

Objectives

579 A vulnerability analysis is performed by the developer to ascertain the presence of “obvious” security vulnerabilities.

580 The objective is to confirm that no identified security vulnerabilities can be exploited in the intended environment for the TOE.

581 An independent vulnerability analysis is performed by the evaluator, which goes beyond the “obvious” security vulnerabilities. The analysis considers the deliverables available for the targeted evaluation assurance level.

582 In addition, the independent vulnerability analysis performed by the evaluator is based on analytical techniques which are employed to discover vulnerabilities that would require sophisticated attackers.

583 The TOE must be shown to be relatively resistant to penetration attack.

Application notes

584 Obvious vulnerabilities are those which are open to exploitation which requires a minimum of understanding of the TOE, skill, technical sophistication, and resources.

585 Independent vulnerability analysis is based on detailed technical information. The attacker is assumed to be thoroughly familiar with the specific implementation of the TOE. The attacker is presumed to have a moderate level of technical sophistication.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.3.1D The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.

AVA_VLA.3.2D The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.3.1C The evidence shall show, for each vulnerability, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.3.2C **The documentation shall justify that the TOE, with the identified vulnerabilities, is relatively resistant to penetration attacks.**

Evaluator action elements:

AVA_VLA.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.3.2E The evaluator shall conduct penetration testing, based on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

AVA_VLA.3.3E The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.3.4E The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of identified vulnerabilities in the target environment.

AVA_VLA.3.5E **The evaluator shall determine that the TOE is relatively resistant to penetration attacks.**

EAL 6
Semiformally verified design and tested

ACM Configuration management

586 Configuration management (CM) is an aspect of establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE. CM systems are put in place to ensure the integrity of the configuration items that they control, by providing a method of tracking these configuration items, and by ensuring that only authorised users are capable of changing them.

ACM_AUT CM automation

Objectives

587 The objective of introducing automated CM tools is to increase the efficiency of the CM system, by simultaneously increasing the reliability of the CM system and reducing the cost of operating it. While both automated and manual CM systems can be bypassed, ignored, or insufficient to prevent unauthorised modification, automated systems are less susceptible to human error or negligence. In addition, while a manual CM system can accomplish all of the same things that an automated system can, manual systems are typically more costly to operate on an ongoing basis.

Application notes

588 For ACM_AUT.1 and ACM_AUT.2, there is a requirement that the automated CM system control changes to the implementation representation of the TOE. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

ACM_AUT.2 Complete CM automation

Objectives

589 In development environments where the configuration items are complex or are being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are performed by authorised developers before their application. It is

the objective of this component to ensure that all configuration items are controlled through automated means.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_AUT.2.1D The developer shall provide a CM plan.

Content and presentation of evidence elements:

ACM_AUT.2.1C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.2.2C The CM plan shall describe how the automated tools are used in the CM system.

ACM_AUT.2.3C **The CM system shall provide an automated means to ensure that only authorised changes are made to the TOE implementation representation, and to all other configuration items.**

ACM_AUT.2.4C The CM system shall provide an automated means to support the generation of any supported TSF from its implementation representation.

ACM_AUT.2.5C The CM system shall provide an automated means to support the comparison of any two supported TSF versions, to ascertain the changes.

ACM_AUT.2.6C **The CM system shall provide an automated means to identify all other configuration items that are affected by the modification of a given configuration item.**

Evaluator action elements:

ACM_AUT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP CM capabilities

Objectives

590 The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TSF from the early design stages through all subsequent maintenance efforts.

591 The objectives of this family include the following:

a) ensuring that the TSF is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items; and

d) enabling recovery to an earlier version of the TOE, in the event that an error occurs through modification, addition, or deletion of TOE configuration items.

Application notes

592 For ACM_CAP.1 and the higher components, there is a requirement that a configuration list be provided. The configuration list contains all configuration items which are maintained by the CM system.

593 For ACM_CAP.2 and the higher components, there is a requirement that the CM documentation include evidence that the CM system is working properly. An example of such evidence might be audit trail output from the CM system. The evaluator is responsible for examining such evidence, to determine that it is sufficient to demonstrate proper functionality of the CM system.

594 For ACM_CAP.2 and the higher components, there is a requirement that evidence be provided that all configuration items are being maintained under the CM system. Since a configuration item refers to an item which is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

595 For ACM_CAP.3 and ACM_CAP.4, there is a requirement that the CM system support the generation of all supported versions of the TOE. This provides the ability to recover to a previous known version in the event that an error occurs through modification, addition or deletion of TOE configuration items.

ACM_CAP.4 Advanced support

Objectives

596 Clear identification of the TOE is required to determine those items under evaluation that are subject to the criteria requirements.

597 Assurance of TOE integrity may be gained by controlling the ability to modify the TOE configuration items. Ensuring proper functionality and use of the CM system also provides assurance that the CM system is correctly enforcing the integrity of the TOE.

598 The ability to generate previous but still supported versions of the TOE is necessary for the resolution of any new flaws discovered during operation.

599 The purpose of acceptance procedures is to confirm that any creation or modification of TSF configuration items is authorised.

600 Integration procedures ensure that the introduction of modifications into the TSF is performed in a controlled and complete manner.

601 Requiring that the CM system be able to identify the master copy of the material used to generate the TSF helps to ensure that the integrity of this material is preserved by the appropriate technical, physical and procedural safeguards.

Dependencies:

ACM_SCP.1 Minimal CM coverage

ALC_DVS.2 Sufficiency of security measures

Developer action elements:

ACM_CAP.4.1D The developer shall use a CM system.

ACM_CAP.4.2D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.4.1C **The CM documentation shall include a configuration list, a CM plan, an acceptance plan, and integration procedures.**

ACM_CAP.4.2C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.3C The CM documentation shall describe the method used to uniquely identify the TOE configuration items.

ACM_CAP.4.4C The CM plan shall describe how the CM system is used.

ACM_CAP.4.5C The CM documentation shall provide evidence that the CM system is working properly.

ACM_CAP.4.6C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.4.7C The CM system shall ensure that only authorised changes are made to the TOE configuration items.

ACM_CAP.4.8C The CM system shall support the generation of all supported versions of the TOE.

ACM_CAP.4.9C The acceptance plan shall describe the procedures used to accept modified or newly created TSF configuration items as part of the TOE.

ACM_CAP.4.10C **The integration procedures shall describe how the CM system is applied in the TOE manufacturing process.**

ACM_CAP.4.11C **The CM system shall require that the person responsible for accepting a configuration item into CM is not the person who developed it.**

ACM_CAP.4.12C **The CM system shall permit clear identification of the TSF.**

ACM_CAP.4.13C **The CM system shall support the audit of all modifications to the TSF, including as a minimum the originator, date, and time in the audit trail.**

ACM_CAP.4.14C **The CM system shall be able to identify the master copy of all material used to generate the TSF.**

ACM_CAP.4.15C **The evidence shall justify that the use of the CM system is sufficient to ensure that only authorised changes are made to the TOE.**

ACM_CAP.4.16C **The evidence shall justify that the integration procedures ensure that the introduction of modifications into the TSF is performed in a controlled and complete manner.**

ACM_CAP.4.17C **The evidence shall justify that the CM system is sufficient to ensure that the person responsible for accepting a configuration item into CM is not the person who developed it.**

ACM_CAP.4.18C **The evidence shall justify that the acceptance procedures provide for an adequate and appropriate review of changes to TSF configuration items.**

Evaluator action elements:

ACM_CAP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_SCP CM scope

Objectives

602 The objective is to ensure that all necessary TOE configuration items are tracked by the CM system. This helps to ensure that the integrity of these configuration items is protected through the capabilities of the CM system.

603 The objectives of this family include the following:

a) ensuring that the TOE implementation representation is tracked;

b) ensuring that all necessary documentation, including problem reports, are tracked during development and operation;

c) ensuring that configuration options (e.g. compiler switches) are tracked; and

d) ensuring that development tools are tracked.

Application notes

604 For ACM_SCP.1 and the higher components, there is a requirement that the TOE implementation representation be tracked by the CM system. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

605 For ACM_SCP.2 and ACM_SCP.3, there is a requirement that security flaws be tracked by the CM system. This requires that information regarding previous security flaws and their resolution be maintained, as well as details regarding current security flaws.

606 For ACM_SCP.3, there is a requirement that development tools and other related information be tracked by the CM system. Examples of development tools are programming languages and compilers. Information pertaining to TOE generation items (such as compiler options, installation/generation options, and build options) is an example of information relating to development tools.

ACM_SCP.3 Development tools CM coverage

Objectives

607 A CM system can control changes only to those items that have been placed under CM. At a minimum, the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation should be placed under CM.

608 The ability to track security flaws under CM ensures that security flaw reports are not lost or forgotten, and allows a developer to track security flaws to their resolution.

609 Development tools play an important role in ensuring the production of a quality version of the TSF. Therefore, it is important to control modifications to these tools.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_SCP.3.1D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_SCP.3.1C As a minimum, the following shall be tracked by the CM system: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation, security flaws, and development tools and related information.

ACM_SCP.3.2C The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator action elements:

ACM_SCP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO Delivery and operation

610 Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

ADO_IGS Installation, generation, and start-up

Objectives

611 Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started in a secure manner as intended by the developer.

Application notes

612 The generation requirements are applicable only to TOEs that provide the ability to generate an operational TOE from source or object code.

613 The installation, generation, and start-up procedures may exist as a separate document, but would typically be grouped with other administrative guidance.

ADO_IGS.1 Installation, generation, and start-up procedures

Dependencies:

AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.1.1D The developer shall document procedures to be used for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.1.1C The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

Evaluator action elements:

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV Development

614 The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. The other family in the development class describes requirements for the internal structure of the TSF.

615 The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, since some of the representations are not necessary for low assurance evaluations.

ADV_FSP Functional specification

Objectives

616 The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is a refinement of the statement of IT functional requirements in the ST of the TOE. The functional specification has to show that all the functional requirements defined in the ST are addressed, and that the TSP is enforced by the TSF.

Application notes

617 In addition to the content indicated in the following requirements, the functional specification shall also include any additional specific detail specified by the documentation notes in the related functional components.

618 The developer must provide evidence that the TSF is completely represented by the functional specification. While a functional specification for the entire TOE would allow an evaluator to determine the TSF boundary, it is not necessary to require that specification when other evidence could be provided to demonstrate the TSF boundary.

619 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the functional specification. In the course of the functional specification evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

620 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

621 While a TSP may represent any policies, TSP models have traditionally represented only subsets of those policies. As a result, the TSP model cannot be treated like every other TSF representation inasmuch as the correspondence between the TSP model to the adjacent abstractions (i.e., TSP and functional specification) may not be complete. As a result, there must be a demonstration of correspondence from the functional specification to the TSP directly, rather than through the intervening representation (i.e., TSP model) where correspondence may be lost. For these reasons, all of the requirements for correspondence between the TSP, TSP model, and functional specification have been included in this family and the correspondence requirements in the Representation correspondence (ADV_RCR) family do not apply to the TSP and TSP model.

622 Beginning with ADV_FSP.1, requirements are defined to ensure that the functional specification is consistent with the TSP. Beginning with ADV_FSP.2, because there is no requirement for a TSP model in ADV_FSP.1, requirements are defined to describe the rules and characteristics of applicable policies of the TSP in the TSP model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g., state transition, non-interference). For example, rules may be represented as “properties” (e.g., simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects”, and “objects”.

623 Since not all policies can be modeled, given the current state of the art, the requirement indicating which policies shall be modeled is subjective. The PP/ST author should identify specific functions and associated policies that are required to be modeled. At the very least, access control policies are expected to be modeled since they are currently within the state of the art.

ADV_FSP.5 Property specification by model interpretation

Application notes

624 The requirement for both an informal and semiformal functional specification is necessary to allow an evaluator to effectively comprehend and evaluate the semiformal representation using the informal representation for support.

Dependencies:

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.5.1D The developer shall provide a functional specification.

ADV_FSP.5.2D The developer shall provide a TSP.

ADV_FSP.5.3D The developer shall provide a formal TSP model.

ADV_FSP.5.4D The developer shall provide a demonstration of correspondence between the formal TSP model and the functional specification.

Content and presentation of evidence elements:

ADV_FSP.5.1C The functional specification shall describe the TSF using both an informal and semiformal style.

ADV_FSP.5.2C The functional specification shall include both an informal and semiformal presentation of syntax, effects, exceptions, error messages, and semantics of all external TSF interfaces.

ADV_FSP.5.3C The functional specification shall include evidence that demonstrates that the TSF is completely represented.

ADV_FSP.5.4C The demonstration of correspondence between the formal TSP model and the functional specification shall describe how the functional specification satisfies the formal TSP model.

ADV_FSP.5.5C The demonstration of correspondence between the formal TSP model and the functional specification shall show that there are no security functions in the functional specification that conflict with the formal TSP model.

ADV_FSP.5.6C The formal TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_FSP.5.7C The formal TSP model shall include a rationale that demonstrates that policies of the TSP that are modeled are satisfied by the formal TSP model.

ADV_FSP.5.8C The formal TSP model shall justify that all policies of the TSP that can be modeled are represented in the formal TSP model.

ADV_FSP.5.9C **The evidence shall justify that the informal and semiformal functional specifications are consistent.**

Evaluator action elements:

ADV_FSP.5.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.5.2E The evaluator shall determine that the functional specification is consistent with the TSP.

ADV_FSP.5.3E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_HLD High-level design

Objectives

625 The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e., subsystems) and relates these units to the functions that they contain. The high-level design provides assurance that the TOE provides an architecture appropriate to implement the claimed functional requirements.

626 The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function and identifies the security functions enforced by the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

Application notes

627 In addition to the content indicated in the following requirements, the high-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

628 The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

629 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the high-level design. In the course of the high-level design evaluation there are essentially three types of evaluator

determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

630 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

631 The term “security functionality” is used to represent operations that a subsystem performs that have some effect on the security functions implemented by the TOE. This distinction is made because design constructs, such as subsystems and modules, do not necessarily relate to specific security functions. While a given subsystem may correspond directly to a security function, or even multiple security functions, it is also possible that many subsystems must be combined to implement a single security function.

632 The term “TSP enforcing subsystems” refers to a subsystem that contributes to the enforcement of the TSP.

ADV_HLD.4 Semiformal high-level explanation

Dependencies:

ADV_FSP.3 Semiformal security policy model

ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements:

ADV_HLD.4.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.4.1C The presentation of the high-level design shall be semiformal.

ADV_HLD.4.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.4.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.4.4C The high-level design shall identify the interfaces of the subsystems of the TSF.

ADV_HLD.4.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.4.6C The high-level design shall describe the separation of the TSF into TSP enforcing and other subsystems.

ADV_HLD.4.7C **The evidence shall justify that the identified means of achieving separation, including any protection mechanisms, are sufficient to ensure a clear and effective separation of TSP enforcing from non-TSP enforcing functions.**

ADV_HLD.4.8C **The evidence shall justify that the TSF mechanisms are sufficient to implement the security functions.**

Evaluator action elements:

ADV_HLD.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.4.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_IMP Implementation representation

Objectives

633 The description of the implementation in the form of source code, firmware, hardware drawings, etc. captures the detailed internal workings of the TSF in support of analysis.

Application notes

634 The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. Source code which is then compiled or a hardware drawing which is used to build the actual hardware are examples of parts of an implementation representation.

635 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the implementation. In the course of the implementation evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a more abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis is necessary. However, since the implementation is the least abstract representation it is likely that further analysis cannot be performed, unless the TSF representations have not been evaluated in a usual order (i.e., most abstract to least abstract). If requirements

are not addressed after the analysis of all TSF representations, this represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

636 In all cases, it is important that the evaluator evaluates the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

637 It is expected that evaluators will use the implementation to directly support other evaluation activities (e.g., vulnerability analysis, test coverage analysis). It is expected that PP/ST authors will select a component that requires that the implementation is complete and comprehensible enough to address the needs of all other requirements included in the PP/ST.

ADV_IMP.3 Structured implementation of the TSF

Dependencies:

ADV_INT.1 Modularity

ADV_LLD.1 Descriptive low-level design

ADV_RCR.1 Informal correspondence demonstration

ALC_TAT.3 Compliance with implementation standards - all parts

Developer action elements:

ADV_IMP.3.1D The developer shall provide the implementation representations for the entire TSF.

Content and presentation of evidence elements:

ADV_IMP.3.1C The implementation representations shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.3.2C The implementation representations shall describe the relationships between all portions of the implementation.

ADV_IMP.3.3C **The implementation representations shall be structured into small and comprehensible sections.**

Evaluator action elements:

ADV_IMP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.3.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_INT TSF internals

Objectives

638 This family of components deals with the internal structure of the TSF. Requirements are established for modularity, the layering of the software architecture to separate levels of abstraction and minimisation of circular dependencies, and the minimisation from the TSF of software that is not TSP enforcing.

639 Modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Thus, a modular design provides the basis for determining the scope of interaction with other elements of the TSF, provides for increased assurance that unexpected effects do not occur, and also provides the basis for designing and evaluating test suites.

640 Design complexity affects how difficult it is to understand the design of the TOE. The simpler the design, the more assurance is gained that there are no hidden vulnerabilities in the design and that the high-level protection requirements are accurately and completely instantiated in the lower level design and the implementation.

641 Design complexity minimisation provides a part of the assurance that the code is understood; the less complex the code in the TSF, the greater the likelihood that the design of the TSF is comprehensible. Design complexity minimisation is a key characteristic of a reference validation mechanism.

Application notes

642 The term “relevant representation” is used in these components to cover the need for an evaluator to check for the appropriate issue (e.g., modularity, complexity) at whichever level of representation (e.g., high-level design, implementation) the requirements are being invoked.

643 The term “portions of the TSF” is used to represent parts of the TSF with a varying granularity based on the available TSF representations. The functional specification allows identification in terms of interfaces, the high-level design allows identification in terms of subsystems, the low-level design allows identification in terms of modules, and the implementation representation allows identification in terms of implementation units (e.g., source code files).

ADV_INT.2 Layering

Application notes

644 This component introduces a reference monitor concept (i.e., small enough to be analysed) by requiring the minimisation of complexity of the portions of the TSF that enforce the access control and information flow policies identified in the TSP.

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

Developer action elements:

ADV_INT.2.1D **The developer shall design and structure the TSF in a modular and layered fashion that avoids unnecessary interactions between the modules of the design, minimises mutual interactions between the layers of the design, and minimises the complexity of the portions of the TSF that enforce any access control and information flow policies.**

ADV_INT.2.2D The developer shall provide an architectural description.

Content and presentation of evidence elements:

ADV_INT.2.1C **The architectural description shall identify the modules of the TSF and the portions of the TSF that enforce any access control and information flow policies.**

ADV_INT.2.2C The architectural description shall describe the purpose, interface, parameters, and effects of each module of the TSF.

ADV_INT.2.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

ADV_INT.2.4C **The architectural description shall describe the layering architecture.**

ADV_INT.2.5C **The architectural description shall show that mutual interactions have been eliminated or minimised, and justify those that remain.**

ADV_INT.2.6C **The architectural description shall describe how the portions of the TSF that enforce any access control and information flow policies have been structured to minimise complexity.**

Evaluator action elements:

ADV_INT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.2.2E The evaluator shall check the relevant representations for compliance with the architectural description.

ADV_LLD Low-level design

Objectives

645 The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined.

646 For each module of the TSF, the low-level design describes its purpose, function, interfaces, dependencies, and the implementation of any TSP enforcing functions.

Application notes

647 In addition to the content indicated in the following requirements, the low-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

648 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the low-level design. In the course of the low-level design evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

649 In all cases, it is important that the evaluator evaluates the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

650 The term “TSP enforcing function” refers to any function that contributes to TSP enforcement. The term “TSP enforcing modules” similarly refers to any module that contributes to TSP enforcement.

ADV_LLD.2 Semiformal low-level design

Dependencies:

ADV_HLD.3 Semiformal high-level design

ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements:

ADV_LLD.2.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements:

ADV_LLD.2.1C **The presentation of the low-level design shall be semiformal.**

ADV_LLD.2.2C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.2.3C The low-level design shall describe the purpose of each module.

ADV_LLD.2.4C The low-level design shall define the interrelationships between the modules in terms of provided functionality and dependencies on other modules.

ADV_LLD.2.5C The low-level design shall describe the implementation of all TSP enforcing functions.

ADV_LLD.2.6C The low-level design shall describe the interfaces of each module in terms of their syntax and semantics.

ADV_LLD.2.7C The low-level design shall provide a demonstration that the TSF is completely represented.

ADV_LLD.2.8C The low-level design shall identify the interfaces of the modules of the TSF visible at the external interface of the TSF.

ADV_LLD.2.9C **The low-level design shall describe the separation of the TSF into TSP enforcing and other modules.**

Evaluator action elements:

ADV_LLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.2.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_RCR Representation correspondence

Objectives

651 The correspondence between the various representations (i.e. functional requirements expressed in the ST, functional specification, high-level design, low-level design, implementation) addresses the correct and complete instantiation of the requirements to the least abstract representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

Application notes

652 The developer must demonstrate to the evaluator that the most detailed, or least abstract, representation of the TSF is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

653 The evaluator must analyse each demonstration of correspondence between abstractions, as well as the results of the analysis of each TSF representation, and then make a determination as to whether the functional requirements in the ST have been satisfied.

654 This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure 5.4, it is intended to address correspondence between the requirements in the ST as well as the TOE summary specification, functional specification, high-level design, low-level design, and implementation representation.

ADV_RCR.2 Semiformal correspondence demonstration

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.2.1D The developer shall provide evidence that the least abstract TSF representation provided is an accurate, consistent, and complete instantiation of the functional requirements expressed in the ST.

Content and presentation of evidence elements:

ADV_RCR.2.1C For each adjacent pair of TSF representations, the evidence shall demonstrate that all parts of the more abstract representation are refined in the less abstract representation.

ADV_RCR.2.2C For each adjacent pair of TSF representations, where portions of both representations are at least semiformally specified, the demonstration of correspondence between those portions of the representations shall be semiformal.

ADV_RCR.2.3C For each adjacent pair of TSF representations, where portions of either representation are informally specified the demonstration of correspondence between those portions of the representations may be informal.

Evaluator action elements:

ADV_RCR.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.2.2E The evaluator shall analyse the correspondence between the functional requirements expressed in the ST and the least abstract representation provided to ensure accuracy, consistency, and completeness.

AGD Guidance documents

655 The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure installation and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

AGD_ADM Administrator guidance

Objectives

656 Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

Application notes

657 The requirements AGD_ADM.1.2C and AGD_ADM.1.11C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

658 The PP/ST author should review the functional components of the PP/ST for guidance on administrator documentation. Those application notes that are relevant to administrator guidance for understanding and proper application of the security functions should be considered for inclusion in the administrator guidance requirements. An example of an administrator guidance document is a reference manual.

AGD_ADM.1 Administrator guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.2C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.3C The administrator guidance shall contain guidelines on the consistent and effective use of the security functions within the TSF.

AGD_ADM.1.4C The administrator guidance shall describe the difference between two types of functions: those which allow an administrator to control security parameters, and those which allow the administrator to obtain information only.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the administrator's control.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall contain guidelines on how the security functions interact.

AGD_ADM.1.8C The administrator guidance shall contain instructions regarding how to configure the TOE.

AGD_ADM.1.9C The administrator guidance shall describe all configuration options that may be used during secure installation of the TOE.

AGD_ADM.1.10C The administrator guidance shall describe details, sufficient for use, of procedures relevant to the administration of security.

AGD_ADM.1.11C The administrator guidance shall be consistent with all other documents supplied for evaluation.

Evaluator action elements:

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_ADM.1.2E The evaluator shall confirm that the installation procedures result in a secure configuration.

AGD_USR User guidance

Objectives

659 User guidance refers to written material that is intended to be used by nonadministrative (human) users of the TOE. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

660 The user guidance provides a basis for assumptions about the use of the TOE and a measure of confidence that non-malicious users and application providers will understand the secure operation of the TOE and will use it as intended.

Application notes

661 The requirement AGD_USR.1.3.C and AGD_USR.1.5C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the user guidance.

662 The PP/ST author should review the functional components of the PP/ST for guidance on user documentation. Those application notes that are relevant to user guidance aimed at the understanding and proper use of the security functions should be considered for inclusion in the user guidance requirements. Examples of user guidance are reference manuals, user guides, and on-line help.

AGD_USR.1 User guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the TSF and interfaces available to the user.

AGD_USR.1.2C The user guidance shall contain guidelines on the use of security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall describe the interaction between user-visible security functions.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation delivered for evaluation.

Evaluator action elements:

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC Life cycle support

663 Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

ALC_DVS Development security

Objectives

664 Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

Application notes

665 The evaluator should decide whether there is a need for visiting the user's site in order to confirm that the requirements of this family are met.

ALC_DVS.2 Sufficiency of security measures

Dependencies:

No dependencies.

Developer action elements:

ALC_DVS.2.1D The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.2.1C The development security documentation shall describe the physical, procedural, personnel, and other security measures that are used to protect the confidentiality and integrity of the TOE during its development.

ALC_DVS.2.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

ALC_DVS.2.3C **The evidence shall justify that the security measures are sufficient to protect the confidentiality and integrity of the TOE.**

Evaluator action elements:

ALC_DVS.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.2.2E The evaluator shall check whether the security measures are being applied.

ALC_LCD Life cycle definition

Objectives

666 Poorly controlled development and maintenance can result in a flawed implementation of a TOE (or a TOE that does not meet all of its security requirements). This, in turn, results in security violations. Therefore, it is important that a model for the development and maintenance of a TOE be established as early as possible in the TOE's life-cycle.

667 Using a model for the development and maintenance of a TOE does not guarantee that the TOE will be free of flaws, nor does it guarantee that the TOE will meet all of its security functional requirements. It is possible that the model chosen was insufficient or inadequate and therefore no benefits in the quality of the TOE could be observed. Using a life-cycle model that has been approved by some group of experts (e.g., academic experts, standards bodies) improves the chances that the development and maintenance models will contribute to the overall quality of the TOE.

Application notes

668 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming relevant after the completion of the evaluation, its evaluation adds assurance through an analysis the life-cycle information for the TOE provided at the time of the evaluation.

669 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE.

670 A standardised life-cycle model is a model that has been approved by some group of experts (e.g., academic experts, standards bodies).

671 A measurable life-cycle model is a model with some arithmetic parameters so that e.g. the coding standards can be measured.

ALC_LCD.2 Standardised life-cycle model

Dependencies:

No dependencies.

Developer action elements:

ALC_LCD.2.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.2.2D The developer shall produce life-cycle definition documentation.

ALC_LCD.2.3D The developer shall use a standardised life-cycle model to develop and maintain the TOE.

Content and presentation of evidence elements:

ALC_LCD.2.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.2.2C The life-cycle definition documentation shall explain why the model was chosen and how it is used to develop and maintain the TOE.

ALC_LCD.2.3C The life-cycle definition documentation shall demonstrate compliance with the standardised life-cycle model.

Evaluator action elements:

ALC_LCD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT Tools and techniques

Objectives

672 Tools and techniques is an aspect of selecting tools which are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to programming languages, documentation, implementation standards, and other parts of the TOE like supporting runtime libraries.

Application notes

673 There is a requirement for well-defined development tools. These are tools which have been shown to be well understood and applicable without the need for intensive further clarification. For example, programming languages and computer

aided design (CAD) systems that are based on an a standard published by standards bodies are considered to be well-defined.

674 Tools and techniques distinguishes between the implementation standards applied by the developer and the implementation standards for “all parts of the TOE” which additionally includes third party software, hardware, or firmware.

675 The requirement in ALC_TAT.1.2C is specifically applicable to programming languages so as to ensure that all statements in the source code have an unambiguous meaning.

ALC_TAT.3 Compliance with implementation standards - all parts

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements:

ALC_TAT.3.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.3.2D The developer shall document the selected implementation dependent options of the development tools.

ALC_TAT.3.3D **The developer shall describe the implementation standards for all parts of the TOE.**

Content and presentation of evidence elements:

ALC_TAT.3.1C Any development tools used for implementation shall be well-defined.

ALC_TAT.3.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

Evaluator action elements:

ALC_TAT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.3.2E The evaluator shall confirm that the implementation standards have been applied.

ATE Tests

676 The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g., functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of the PP/ ST. Testing provides assurance that the TSF satisfies at least the security functional

requirements, although it cannot establish that the TSF does no more than what was specified. Testing may also be directed toward the internals of the TSF, such as the testing of subsystems and modules against their specifications.

677 The aspects of coverage and depth have been separated from functional tests for reasons of increased flexibility in applying the components of the families. However, the requirements in these three families are intended to be applied together.

678 The independent testing has dependencies on the other families to provide the necessary information to support the requirements, but is primarily concerned with independent evaluator actions.

679 This class does not address penetration testing, which is directed toward finding vulnerabilities that enable a user to violate the security policy. Penetration testing is addressed separately as an aspect of vulnerability assessment in the class AVA.

ATE_COV Coverage

Objectives

680 This family addresses those aspects of testing that deal with completeness of testing. That is, it addresses the extent to which the TOE security functions are tested, whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified, and whether or not the order in which testing proceeds correctly accounts for functional dependencies between the portions of the TOE being tested.

Application notes

681 The specific documentation required by the coverage components will be determined, in most cases, by the documentation stipulated in the level of ATE_FUN that is specified. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_COV.3 Ordered testing

Objectives

682 The objective is that testing completely address the security functions.

683 The objective is to ensure that there is a detailed correspondence between the tests and the security functions.

684 In this component, an additional objective is detailed justification that testing is structured such as to avoid circular arguments about the correctness of the portions of the TOE being tested.

Application notes

685 Ordering dependencies between tests can be of different forms e.g., test A provides a result to test B; test A cannot run before test B, since it breaks something required by test B; test failure in test B might be because of a failure in “untested” test A.

Dependencies:

ADV_FSP.1 TOE and security policy

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.3.1D The developer shall provide an analysis of the test coverage.

ATE_COV.3.2D **The developer shall provide an analysis of ordering dependencies of tests.**

Content and presentation of evidence elements:

ATE_COV.3.1C The analysis of the test coverage shall demonstrate that the tests identified in the test documentation cover the TSF.

ATE_COV.3.2C The analysis of the test coverage shall demonstrate the correspondence between the security functions and the tests identified in the test documentation.

ATE_COV.3.3C **The analysis documentation shall justify that the correspondence is complete.**

ATE_COV.3.4C **The analysis documentation shall describe the ordering dependencies of tests.**

ATE_COV.3.5C **The analysis documentation shall justify that the test plans and procedures are consistent with the ordering dependencies of tests.**

Evaluator action elements:

ATE_COV.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT Depth

Objectives

686 The components in this family deal with the level of detail to which the TOE is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.

687 The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internals of the TOE, are more likely to discover any malicious code that has been inserted.

Application notes

688 The specific amount and type of documentation and evidence will, in general, be determined by that required by level of ATE_FUN selected. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_DPT.3 Testing - low level design

Objectives

689 The functional specification of a TOE provides a high level description of the external workings of the TSF. Testing at the level of the functional specification, in order to demonstrate the presence of any flaws, provides assurance that the TSF functional specification has been correctly realised.

690 The subsystems of a TOE provide a high level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

691 The modules of a TOE provide a description of the internal workings of the TSF. Testing at the level of the modules, in order to demonstrate the presence of any flaws, provides assurance that the TSF modules have been correctly realised.

Application notes

692 The functional specification representation is used to express the notion of the most abstract representation of the TSF.

693 The developer is expected to describe the testing of the high level design of the TSF in terms of “subsystems”. The term “subsystem” is used to express the notion of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar notion of decomposition.

694 The developer is expected to describe the testing of the low level design of the TSF in terms of “modules”. The term “modules” is used to express the notion of decomposing each of the “subsystems” of the TSF into a relatively small number of parts. While the developer is not required to actually have “modules”, the developer is expected to represent a similar notion of decomposition.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ADV_LLD.1 Descriptive low-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.3.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.3.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TOE operates in accordance with the functional specification, high level design, and low level design of the TSF.

Evaluator action elements:

ATE_DPT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_FUN Functional tests

Objectives

695 Functional testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family “Functional tests” is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through testing.

696 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

Application notes

697 Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures should also show how the test results is derived from the test inputs.

698 The developer shall eliminate all security relevant flaws discovered during testing.

699 The developer shall test the TSF to determine that no new security relevant flaws have been introduced as a result of eliminating discovered security relevant flaws.

ATE_FUN.1 Functional testing

Objectives

700 The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

Dependencies:

ATE_COV.1 Complete coverage - informal

ATE_DPT.1 Testing - functional specification

Developer action elements:

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, and test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function.

ATE_FUN.1.4C The test results in the test documentation shall show the expected results of each test.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each security function operates as specified.

Evaluator action elements:

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND Independent testing

Objectives

701 The objective is to demonstrate that the security functions perform as specified.

702 Additionally, an objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer which results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

Application notes

703 The testing specified in this family can be performed by a party other than the evaluator (e.g., an independent laboratory, an objective consumer organisation).

704 This family deals with the degree to which there is independent functional testing of the TOE. Independent functional testing may take the form of repeating the developer's functional tests, in whole or in part. It may also take the form of the augmentation of the developer's functional tests, either to extend the scope or the depth of the developer's tests.

ATE_IND.2 Independent testing - sample

Objectives

705 The objective is to demonstrate that the security functions perform as specified.

706 In this component, the objective is to select and repeat a sample of the developer testing.

Application notes

707 The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required to run tests. The need for documentation is supported by the dependencies to other assurance families.

708 Additionally, suitability of the TOE for testing may be based on other considerations e.g., the version of the TOE submitted by the developer is not the final version.

709 The developer is required to perform testing and to provide test documentation and test results. This is addressed by the ATE_FUN family.

710 Testing may be selective and shall be based upon all available documentation.

Dependencies:

ADV_FSP.1 TOE and security policy

AGD_USR.1 User guidance

AGD_ADM.1 Administrator guidance

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.2.1C The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test the TSF to confirm that the TSF operates as specified.

ATE_IND.2.3E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

AVA Vulnerability assessment

711 The class “Vulnerability assessment” encompasses four families: covert channel analysis (AVA_CCA), misuse (AVA_MSU), strength of TOE security functions (AVA_SOF) and vulnerability analysis (AVA_VLA). The class addresses the existence of exploitable covert channels, the misuse or incorrect configuration of the TOE, the ability for all critical security mechanisms to withstand direct attack and the definition and assessment of penetration tests to exploit vulnerabilities introduced in the development or the operation of the TOE.

AVA_CCA Covert channel analysis

Objectives

712 Covert channel analysis is carried out to determine the existence and potential capacity of unintended signalling channels that may be exploited by malicious code.

713 The assurance requirements address the threat that unintended and exploitable signalling paths exist which may be exercised to violate the security policy.

Application notes

714 Channel capacity estimations are based upon informal engineering measurements, as well as actual test measurements.

715 Details of the assumptions upon which the covert channel analysis is based shall be given, e.g., processor speed, configuration, memory, and cache size.

716 Test parameters details are (e.g., processor speed, memory and cache size), relevant configuration parameters, how the channel was exercised, used to obtain the capacity during testing.

717 The selective validation of the covert channel analysis through testing allows the evaluator the opportunity to verify any aspect of the covert channel analysis (e.g., identification, capacity estimation, elimination, monitoring, and exploitation scenarios). This does not impose a requirement to demonstrate the entire set of covert channel analysis results.

718 If there are no information flow control policies in the ST, this family of assurance requirements is no longer applicable since this family only applies to information flow control policies. Even if there are no specific functional requirements (e.g.,

FDP_IFF.1 to FDP_IFF.3) for eliminating, limiting, or monitoring covert channels, this family still requires the identification of covert channels.

AVA_CCA.2 Systematic covert channel analysis

Objectives

719 The objective is to identify covert channels which are identifiable through analysis.

720 In this component, the objective is to perform a systematic search for covert channels.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_IMP.2 Implementation of the TSF

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_CCA.2.1D The developer shall conduct a search for covert channels for each information flow control policy.

AVA_CCA.2.2D The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements:

AVA_CCA.2.1C The analysis documentation shall identify covert channels.

AVA_CCA.2.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA.2.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA.2.4C The analysis documentation shall describe the method used for estimating channel capacity, which shall be based on worst case scenarios.

AVA_CCA.2.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_CCA.2.6C **The analysis documentation shall provide evidence that the method used to identify covert channels is systematic.**

Evaluator action elements:

AVA_CCA.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA.2.2E The evaluator shall confirm that the results of the covert channels analysis meet the functional requirements.

AVA_CCA.2.3E The evaluator shall selectively validate the covert channel analysis through testing.

AVA_MSU Misuse

Objectives

721 Misuse investigates whether the TOE can be configured or used in a manner which is insecure but which an administrator or end-user of the TOE would reasonably believe to be secure.

722 The objective is to minimise the risk of human or other errors in operation which may deactivate, disable, or fail to activate security functions.

723 The objective is to minimise the probability of configuring or installing the TOE in a way which is insecure, without the end user or administrator being able to recognise it.

Application notes

724 Conflicting, misleading or incomplete guidance may result in a user of the TOE believing that the TOE is secure, when it is not. Conflicting guidance can result in vulnerabilities.

725 An example of conflicting guidance would be two guidance instructions which imply different outcomes when the same input is supplied.

726 An example of misleading guidance would be the description of a single guidance instruction which could be parsed in more than one way, one of which may result in an insecure state.

727 An example of completeness would be referencing assertions of dependencies on external security measures e.g., such as external procedural, physical and personnel controls.

AVA_MSU.2 Misuse analysis - independent verification

Objectives

728 The objective is to ensure that conflicting guidance in the guidance documentation have been addressed.

729 In this component, the objective is to provide additional assurance by performing an independent analysis.

Dependencies:

ADO_IGS.1 Installation, generation, and start-up procedures

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.2.1D The developer shall document an analysis of the guidance documentation for conflicting and incomplete guidance.

AVA_MSU.2.2D The developer shall ensure that the guidance documentation contains no misleading or unreasonable guidance.

Content and presentation of evidence elements:

AVA_MSU.2.1C The analysis documentation shall provide a rationale that demonstrates that the guidance is not conflicting and is complete.

Evaluator action elements:

AVA_MSU.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E The evaluator shall determine that there is no misleading or unreasonable guidance in the guidance documentation.

AVA_MSU.2.3E The evaluator shall repeat any procedures in the guidance documentation to ensure that they produce the documented results.

AVA_MSU.2.4E The evaluator shall perform independent testing to confirm that the TOE can be configured and operated securely using only the guidance documentation.

AVA_SOF Strength of TOE security functions

Objectives

730 Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security functions claim.

Application notes

731 Security functions are implemented by security mechanisms. For example, a password mechanism can be used in the implementation of the identification and authentication security function.

732 The strength of TOE security functions evaluation is performed at the level of the security mechanism, but its results provide knowledge about the ability of the related security function to counter the identified threats.

733 The strength of a function is rated ‘basic’ if the analysis shows that the function provides adequate protection against unintended or casual breach of TOE security by attackers possessing a low attack potential.

734 The strength of a function is rated ‘medium’ if the analysis shows that the function provides adequate protection against attackers possessing a moderate attack potential.

735 The strength of a function is rated ‘high’ if the analysis shows that the function provides adequate protection against attackers possessing a high attack potential.

736 The attack potential is derived from the attacker’s expertise, opportunities, resources, and motivation.

AVA_SOF.1 Strength of TOE security function evaluation

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

Developer action elements:

AVA_SOF.1.1D The developer shall identify all TOE security mechanisms for which a strength of TOE security function analysis is appropriate.

AVA_SOF.1.2D The developer shall perform a strength of TOE security function analysis for each identified mechanism.

Content and presentation of evidence elements:

AVA_SOF.1.1C The strength of TOE security function analysis shall determine the impact of the identified TOE security mechanisms on the ability of the TOE security functions to counter the threats.

AVA_SOF.1.2C The strength of TOE security function analysis shall demonstrate that the identified strength of the security functions is consistent with the security objectives of the TOE.

AVA_SOF.1.3C Each strength claim shall be either basic, medium, or high.

Evaluator action elements:

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that all TOE security mechanisms requiring a strength analysis have been identified.

AVA_SOF.1.3E The evaluator shall confirm that the strength claims are correct.

AVA_VLA Vulnerability analysis

Objectives

737 Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or e.g., by flaw hypotheses, could allow malicious users to violate the TSP.

738 Vulnerability analysis deals with the threats that a malicious user will be able to discover flaws that will allow access to resources (e.g., data), allow the ability to interfere with or alter the TSF, or interfere with the authorised capabilities of other users.

Application notes

739 The vulnerability analysis should consider the contents of all the TOE deliverables for the targeted evaluation assurance level.

740 Obvious vulnerabilities are those that allow common attacks or those that might be suggested by the TOE interface description. Obvious vulnerabilities are those in the public domain, details of which should be known to a developer or available from an evaluation oversight body.

741 The evidence identifies all the TOE documentation upon which the search for flaws was based.

AVA_VLA.4 Highly resistant

Objectives

742 A vulnerability analysis is performed by the developer to ascertain the presence of “obvious” security vulnerabilities.

743 The objective is to confirm that no identified security vulnerabilities can be exploited in the intended environment for the TOE.

744 An independent vulnerability analysis is performed by the evaluator, which goes beyond the “obvious” security vulnerabilities. The analysis considers the deliverables available for the targeted evaluation assurance level.

745 In addition, the independent vulnerability analysis performed by the evaluator is based on analytical techniques which are employed to discover vulnerabilities that would require sophisticated attackers.

746 The TOE must be shown to be highly resistant to penetration attacks.

Application notes

747 Obvious vulnerabilities are those which are open to exploitation which requires a minimum of understanding of the TOE, skill, technical sophistication, and resources.

748 Independent vulnerability analysis is based on highly detailed technical information. The attacker is assumed to be thoroughly familiar with the specific implementation of the TOE. The attacker is presumed to have a high level of technical sophistication.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.4.1D The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.

AVA_VLA.4.2D The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.4.1C The evidence shall show, for each vulnerability, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.4.2C **The documentation shall justify that the TOE, with the identified vulnerabilities, is highly resistant to penetration attacks.**

AVA_VLA.4.3C **The analysis documentation shall provide a justification that the analysis completely addresses the TOE deliverables.**

Evaluator action elements:

AVA_VLA.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.4.2E The evaluator shall conduct penetration testing, based on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

AVA_VLA.4.3E The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.4.4E The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of identified vulnerabilities in the target environment.

AVA_VLA.4.5E **The evaluator shall determine that the TOE is highly resistant to penetration attacks.**

EAL 7
Formally verified design and tested

ACM Configuration management

749 Configuration management (CM) is an aspect of establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE. CM systems are put in place to ensure the integrity of the configuration items that they control, by providing a method of tracking these configuration items, and by ensuring that only authorised users are capable of changing them.

ACM_AUT CM automation

Objectives

750 The objective of introducing automated CM tools is to increase the efficiency of the CM system, by simultaneously increasing the reliability of the CM system and reducing the cost of operating it. While both automated and manual CM systems can be bypassed, ignored, or insufficient to prevent unauthorised modification, automated systems are less susceptible to human error or negligence. In addition, while a manual CM system can accomplish all of the same things that an automated system can, manual systems are typically more costly to operate on an ongoing basis.

Application notes

751 For ACM_AUT.1 and ACM_AUT.2, there is a requirement that the automated CM system control changes to the implementation representation of the TOE. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

ACM_AUT.2 Complete CM automation

Objectives

752 In development environments where the configuration items are complex or are being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are performed by authorised developers before their application. It is

the objective of this component to ensure that all configuration items are controlled through automated means.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_AUT.2.1D The developer shall provide a CM plan.

Content and presentation of evidence elements:

ACM_AUT.2.1C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.2.2C The CM plan shall describe how the automated tools are used in the CM system.

ACM_AUT.2.3C The CM system shall provide an automated means to ensure that only authorised changes are made to the TOE implementation representation, and to all other configuration items.

ACM_AUT.2.4C The CM system shall provide an automated means to support the generation of any supported TSF from its implementation representation.

ACM_AUT.2.5C The CM system shall provide an automated means to support the comparison of any two supported TSF versions, to ascertain the changes.

ACM_AUT.2.6C The CM system shall provide an automated means to identify all other configuration items that are affected by the modification of a given configuration item.

Evaluator action elements:

ACM_AUT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP CM capabilities

Objectives

753 The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TSF from the early design stages through all subsequent maintenance efforts.

754 The objectives of this family include the following:

a) ensuring that the TSF is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items; and

d) enabling recovery to an earlier version of the TOE, in the event that an error occurs through modification, addition, or deletion of TOE configuration items.

Application notes

755 For ACM_CAP.1 and the higher components, there is a requirement that a configuration list be provided. The configuration list contains all configuration items which are maintained by the CM system.

756 For ACM_CAP.2 and the higher components, there is a requirement that the CM documentation include evidence that the CM system is working properly. An example of such evidence might be audit trail output from the CM system. The evaluator is responsible for examining such evidence, to determine that it is sufficient to demonstrate proper functionality of the CM system.

757 For ACM_CAP.2 and the higher components, there is a requirement that evidence be provided that all configuration items are being maintained under the CM system. Since a configuration item refers to an item which is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

758 For ACM_CAP.3 and ACM_CAP.4, there is a requirement that the CM system support the generation of all supported versions of the TOE. This provides the ability to recover to a previous known version in the event that an error occurs through modification, addition or deletion of TOE configuration items.

ACM_CAP.4 Advanced support

Objectives

759 Clear identification of the TOE is required to determine those items under evaluation that are subject to the criteria requirements.

760 Assurance of TOE integrity may be gained by controlling the ability to modify the TOE configuration items. Ensuring proper functionality and use of the CM system also provides assurance that the CM system is correctly enforcing the integrity of the TOE.

761 The ability to generate previous but still supported versions of the TOE is necessary for the resolution of any new flaws discovered during operation.

762 The purpose of acceptance procedures is to confirm that any creation or modification of TSF configuration items is authorised.

763 Integration procedures ensure that the introduction of modifications into the TSF is performed in a controlled and complete manner.

764 Requiring that the CM system be able to identify the master copy of the material used to generate the TSF helps to ensure that the integrity of this material is preserved by the appropriate technical, physical and procedural safeguards.

Dependencies:

ACM_SCP.1 Minimal CM coverage

ALC_DVS.2 Sufficiency of security measures

Developer action elements:

ACM_CAP.4.1D The developer shall use a CM system.

ACM_CAP.4.2D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.4.1C The CM documentation shall include a configuration list, a CM plan, an acceptance plan, and integration procedures.

ACM_CAP.4.2C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.3C The CM documentation shall describe the method used to uniquely identify the TOE configuration items.

ACM_CAP.4.4C The CM plan shall describe how the CM system is used.

ACM_CAP.4.5C The CM documentation shall provide evidence that the CM system is working properly.

ACM_CAP.4.6C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.4.7C The CM system shall ensure that only authorised changes are made to the TOE configuration items.

ACM_CAP.4.8C The CM system shall support the generation of all supported versions of the TOE.

ACM_CAP.4.9C The acceptance plan shall describe the procedures used to accept modified or newly created TSF configuration items as part of the TOE.

ACM_CAP.4.10C The integration procedures shall describe how the CM system is applied in the TOE manufacturing process.

ACM_CAP.4.11C The CM system shall require that the person responsible for accepting a configuration item into CM is not the person who developed it.

ACM_CAP.4.12C The CM system shall permit clear identification of the TSF.

ACM_CAP.4.13C The CM system shall support the audit of all modifications to the TSF, including as a minimum the originator, date, and time in the audit trail.

ACM_CAP.4.14C The CM system shall be able to identify the master copy of all material used to generate the TSF.

ACM_CAP.4.15C The evidence shall justify that the use of the CM system is sufficient to ensure that only authorised changes are made to the TOE.

ACM_CAP.4.16C The evidence shall justify that the integration procedures ensure that the introduction of modifications into the TSF is performed in a controlled and complete manner.

ACM_CAP.4.17C The evidence shall justify that the CM system is sufficient to ensure that the person responsible for accepting a configuration item into CM is not the person who developed it.

ACM_CAP.4.18C The evidence shall justify that the acceptance procedures provide for an adequate and appropriate review of changes to TSF configuration items.

Evaluator action elements:

ACM_CAP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_SCP CM scope

Objectives

765 The objective is to ensure that all necessary TOE configuration items are tracked by the CM system. This helps to ensure that the integrity of these configuration items is protected through the capabilities of the CM system.

766 The objectives of this family include the following:

a) ensuring that the TOE implementation representation is tracked;

b) ensuring that all necessary documentation, including problem reports, are tracked during development and operation;

c) ensuring that configuration options (e.g. compiler switches) are tracked; and

d) ensuring that development tools are tracked.

Application notes

767 For ACM_SCP.1 and the higher components, there is a requirement that the TOE implementation representation be tracked by the CM system. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code, but in other TOEs the implementation representation may refer to a combination of software, hardware, and firmware.

768 For ACM_SCP.2 and ACM_SCP.3, there is a requirement that security flaws be tracked by the CM system. This requires that information regarding previous security flaws and their resolution be maintained, as well as details regarding current security flaws.

769 For ACM_SCP.3, there is a requirement that development tools and other related information be tracked by the CM system. Examples of development tools are programming languages and compilers. Information pertaining to TOE generation items (such as compiler options, installation/generation options, and build options) is an example of information relating to development tools.

ACM_SCP.3 Development tools CM coverage

Objectives

770 A CM system can control changes only to those items that have been placed under CM. At a minimum, the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation should be placed under CM.

771 The ability to track security flaws under CM ensures that security flaw reports are not lost or forgotten, and allows a developer to track security flaws to their resolution.

772 Development tools play an important role in ensuring the production of a quality version of the TSF. Therefore, it is important to control modifications to these tools.

Dependencies:

ACM_CAP.2 Authorisation controls

Developer action elements:

ACM_SCP.3.1D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_SCP.3.1C As a minimum, the following shall be tracked by the CM system: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation, security flaws, and development tools and related information.

ACM_SCP.3.2C The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator action elements:

ACM_SCP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO Delivery and operation

773 Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

ADO_IGS Installation, generation, and start-up

Objectives

774 Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started in a secure manner as intended by the developer.

Application notes

775 The generation requirements are applicable only to TOEs that provide the ability to generate an operational TOE from source or object code.

776 The installation, generation, and start-up procedures may exist as a separate document, but would typically be grouped with other administrative guidance.

ADO_IGS.1 Installation, generation, and start-up procedures

Dependencies:

AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.1.1D The developer shall document procedures to be used for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.1.1C The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

Evaluator action elements:

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV Development

777 The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. The other family in the development class describes requirements for the internal structure of the TSF.

778 The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, since some of the representations are not necessary for low assurance evaluations.

ADV_FSP Functional specification

Objectives

779 The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is a refinement of the statement of IT functional requirements in the ST of the TOE. The functional specification has to show that all the functional requirements defined in the ST are addressed, and that the TSP is enforced by the TSF.

Application notes

780 In addition to the content indicated in the following requirements, the functional specification shall also include any additional specific detail specified by the documentation notes in the related functional components.

781 The developer must provide evidence that the TSF is completely represented by the functional specification. While a functional specification for the entire TOE would allow an evaluator to determine the TSF boundary, it is not necessary to require that specification when other evidence could be provided to demonstrate the TSF boundary.

782 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the functional specification. In the course of the functional specification evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

783 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

784 While a TSP may represent any policies, TSP models have traditionally represented only subsets of those policies. As a result, the TSP model cannot be treated like every other TSF representation inasmuch as the correspondence between the TSP model to the adjacent abstractions (i.e., TSP and functional specification) may not be complete. As a result, there must be a demonstration of correspondence from the functional specification to the TSP directly, rather than through the intervening representation (i.e., TSP model) where correspondence may be lost. For these reasons, all of the requirements for correspondence between the TSP, TSP model, and functional specification have been included in this family and the correspondence requirements in the Representation correspondence (ADV_RCR) family do not apply to the TSP and TSP model.

785 Beginning with ADV_FSP.1, requirements are defined to ensure that the functional specification is consistent with the TSP. Beginning with ADV_FSP.2, because there is no requirement for a TSP model in ADV_FSP.1, requirements are defined to describe the rules and characteristics of applicable policies of the TSP in the TSP model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g., state transition, non-interference). For example, rules may be represented as “properties” (e.g., simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects”, and “objects”.

786 Since not all policies can be modeled, given the current state of the art, the requirement indicating which policies shall be modeled is subjective. The PP/ST author should identify specific functions and associated policies that are required to be modeled. At the very least, access control policies are expected to be modeled since they are currently within the state of the art.

ADV_FSP.6 Formal specification of the TSF properties

Application notes

787 The requirement for both an informal and formal functional specification is necessary to allow an evaluator to effectively comprehend and evaluate the more formal representation using the informal representation for support.

Dependencies:

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.6.1D The developer shall provide a functional specification.

ADV_FSP.6.2D The developer shall provide a TSP.

ADV_FSP.6.3D The developer shall provide a formal TSP model.

ADV_FSP.6.4D **The developer shall provide a proof of correspondence between the formal TSP model and the functional specification.**

Content and presentation of evidence elements:

ADV_FSP.6.1C **The functional specification shall describe the TSF using both an informal and formal style.**

ADV_FSP.6.2C **The functional specification shall include both an informal and formal presentation of syntax, effects, exceptions, error messages, and semantics of all external TSF interfaces.**

ADV_FSP.6.3C The functional specification shall include evidence that demonstrates that the TSF is completely represented.

ADV_FSP.6.4C **The proof of correspondence between the formal TSP model and the functional specification shall demonstrate that the functional specification satisfies the formal TSP model.**

ADV_FSP.6.5C **The proof of correspondence between the formal TSP model and the functional specification shall demonstrate that there are no security functions in the functional specification that conflict with the formal TSP model.**

ADV_FSP.6.6C The formal TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_FSP.6.7C The formal TSP model shall include a rationale that demonstrates that policies of the TSP that are modeled are satisfied by the formal TSP model.

ADV_FSP.6.8C The formal TSP model shall justify that all policies of the TSP that can be modeled are represented in the formal TSP model.

ADV_FSP.6.9C **The evidence shall justify that the informal and formal functional specifications are consistent.**

Evaluator action elements:

ADV_FSP.6.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.6.2E The evaluator shall determine that the functional specification is consistent with the TSP.

ADV_FSP.6.3E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_HLD High-level design

Objectives

788 The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e., subsystems) and relates these units to the functions that they contain. The high-level design provides assurance that the TOE provides an architecture appropriate to implement the claimed functional requirements.

789 The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function and identifies the security functions enforced by the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

Application notes

790 In addition to the content indicated in the following requirements, the high-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

791 The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

792 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the high-level design. In the course of the high-level design evaluation there are essentially three types of evaluator

determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

793 In all cases, it is important that the evaluator evaluate the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

794 The term “security functionality” is used to represent operations that a subsystem performs that have some effect on the security functions implemented by the TOE. This distinction is made because design constructs, such as subsystems and modules, do not necessarily relate to specific security functions. While a given subsystem may correspond directly to a security function, or even multiple security functions, it is also possible that many subsystems must be combined to implement a single security function.

795 The term “TSP enforcing subsystems” refers to a subsystem that contributes to the enforcement of the TSP.

ADV_HLD.5 Formal high-level design

Dependencies:

ADV_FSP.4 Formal security policy model

ADV_RCR.3 Formal correspondence demonstration

Developer action elements:

ADV_HLD.5.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.5.1C **The presentation of the high-level design shall be formal.**

ADV_HLD.5.2C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.5.3C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.5.4C The high-level design shall identify the interfaces of the subsystems of the TSF.

ADV_HLD.5.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.5.6C The high-level design shall describe the separation of the TSF into TSP enforcing and other subsystems.

ADV_HLD.5.7C The evidence shall justify that the identified means of achieving separation, including any protection mechanisms, are sufficient to ensure a clear and effective separation of TSP enforcing from non-TSP enforcing functions.

ADV_HLD.5.8C The evidence shall justify that the TSF mechanisms are sufficient to implement the security functions.

Evaluator action elements:

ADV_HLD.5.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.5.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_IMP Implementation representation

Objectives

796 The description of the implementation in the form of source code, firmware, hardware drawings, etc. captures the detailed internal workings of the TSF in support of analysis.

Application notes

797 The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. Source code which is then compiled or a hardware drawing which is used to build the actual hardware are examples of parts of an implementation representation.

798 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the implementation. In the course of the implementation evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a more abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis is necessary. However, since the implementation is the least abstract representation it is likely that further analysis cannot be performed, unless the TSF representations have not been evaluated in a usual order (i.e., most abstract to least abstract). If requirements

are not addressed after the analysis of all TSF representations, this represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

799 In all cases, it is important that the evaluator evaluates the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

800 It is expected that evaluators will use the implementation to directly support other evaluation activities (e.g., vulnerability analysis, test coverage analysis). It is expected that PP/ST authors will select a component that requires that the implementation is complete and comprehensible enough to address the needs of all other requirements included in the PP/ST.

ADV_IMP.3 Structured implementation of the TSF

Dependencies:

ADV_INT.1 Modularity

ADV_LLD.1 Descriptive low-level design

ADV_RCR.1 Informal correspondence demonstration

ALC_TAT.3 Compliance with implementation standards - all parts

Developer action elements:

ADV_IMP.3.1D The developer shall provide the implementation representations for the entire TSF.

Content and presentation of evidence elements:

ADV_IMP.3.1C The implementation representations shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.3.2C The implementation representations shall describe the relationships between all portions of the implementation.

ADV_IMP.3.3C The implementation representations shall be structured into small and comprehensible sections.

Evaluator action elements:

ADV_IMP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.3.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_INT TSF internals

Objectives

801 This family of components deals with the internal structure of the TSF. Requirements are established for modularity, the layering of the software architecture to separate levels of abstraction and minimisation of circular dependencies, and the minimisation from the TSF of software that is not TSP enforcing.

802 Modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Thus, a modular design provides the basis for determining the scope of interaction with other elements of the TSF, provides for increased assurance that unexpected effects do not occur, and also provides the basis for designing and evaluating test suites.

803 Design complexity affects how difficult it is to understand the design of the TOE. The simpler the design, the more assurance is gained that there are no hidden vulnerabilities in the design and that the high-level protection requirements are accurately and completely instantiated in the lower level design and the implementation.

804 Design complexity minimisation provides a part of the assurance that the code is understood; the less complex the code in the TSF, the greater the likelihood that the design of the TSF is comprehensible. Design complexity minimisation is a key characteristic of a reference validation mechanism.

Application notes

805 The term “relevant representation” is used in these components to cover the need for an evaluator to check for the appropriate issue (e.g., modularity, complexity) at whichever level of representation (e.g., high-level design, implementation) the requirements are being invoked.

806 The term “portions of the TSF” is used to represent parts of the TSF with a varying granularity based on the available TSF representations. The functional specification allows identification in terms of interfaces, the high-level design allows identification in terms of subsystems, the low-level design allows identification in terms of modules, and the implementation representation allows identification in terms of implementation units (e.g., source code files).

ADV_INT.3 Minimisation of Complexity

Application notes

807 This component requires that the reference monitor property “small enough to be analysed” is fully addressed. When this component is combined with the functional requirements FPT_RVM.1 and FPT_SEP.3, the reference monitor concept would be fully realised.

Dependencies:

ADV_IMP.2 Implementation of the TSF

ADV_LLD.1 Descriptive low-level design

Developer action elements:

ADV_INT.3.1D **The developer shall design and structure the TSF in a modular and layered fashion that avoids unnecessary interactions between the modules of the design, minimises mutual interactions between the layers of the design, and minimises the complexity of the entire TSF.**

ADV_INT.3.2D The developer shall provide an architectural description.

ADV_INT.3.3D **The developer shall design and structure the portions of the TSF that enforce any access control and information flow policies such that they are small enough to be analysed.**

ADV_INT.3.4D **The developer shall ensure that functions that are not relevant to TSP enforcement are excluded from the TSF.**

Content and presentation of evidence elements:

ADV_INT.3.1C The architectural description shall identify the modules of the TSF and the portions of the TSF that enforce any access control and information flow policies.

ADV_INT.3.2C The architectural description shall describe the purpose, interface, parameters, and side-effects of each module of the TSF.

ADV_INT.3.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

ADV_INT.3.4C The architectural description shall describe the layering architecture.

ADV_INT.3.5C The architectural description shall show that mutual interactions have been eliminated or minimised, and justify those that remain.

ADV_INT.3.6C **The architectural description shall describe how the entire TSF has been structured to minimise complexity.**

ADV_INT.3.7C **The architectural description shall justify the inclusion of any non TSP enforcing modules in the TSF.**

Evaluator action elements:

ADV_INT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.3.2E The evaluator shall check the relevant representations for compliance with the architectural description.

ADV_INT.3.3E **The evaluator shall confirm that the portions of the TSF that enforce any access control and information flow policies are small enough to be analysed.**

ADV_LLD Low-level design

Objectives

808 The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined.

809 For each module of the TSF, the low-level design describes its purpose, function, interfaces, dependencies, and the implementation of any TSP enforcing functions.

Application notes

810 In addition to the content indicated in the following requirements, the low-level design shall also include any additional specific detail specified by the documentation notes in the related functional components.

811 The evaluator of the TOE is expected to make determinations regarding the functional requirements in the ST relevant to the low-level design. In the course of the low-level design evaluation there are essentially three types of evaluator determination: specific functional requirements are met and no further work (e.g., with a less abstract representation of the TSF) is necessary; specific functional requirements are violated and the TOE fails to meet its requirements; and specific functional requirements have not been addressed and further analysis (of another TSF representation) is necessary. Whenever more analysis is necessary, the evaluator is expected to carry that information forward to the analysis of other TSF representations. If requirements are not addressed after the analysis of the last provided TSF representation, this also represents a failure of the TOE evaluation. Note that this more comprehensive failure determination requirement is realised in the Representation correspondence (ADV_RCR) family.

812 In all cases, it is important that the evaluator evaluates the TSF as a unit since in many cases the security functions must cooperate to meet specific functional requirements and also each security function must not interfere with the operation of any other security function.

813 The term “TSP enforcing function” refers to any function that contributes to TSP enforcement. The term “TSP enforcing modules” similarly refers to any module that contributes to TSP enforcement.

ADV_LLD.2 Semiformal low-level design

Dependencies:

ADV_HLD.3 Semiformal high-level design

ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements:

ADV_LLD.2.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements:

ADV_LLD.2.1C The presentation of the low-level design shall be semiformal.

ADV_LLD.2.2C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.2.3C The low-level design shall describe the purpose of each module.

ADV_LLD.2.4C The low-level design shall define the interrelationships between the modules in terms of provided functionality and dependencies on other modules.

ADV_LLD.2.5C The low-level design shall describe the implementation of all TSP enforcing functions.

ADV_LLD.2.6C The low-level design shall describe the interfaces of each module in terms of their syntax and semantics.

ADV_LLD.2.7C The low-level design shall provide a demonstration that the TSF is completely represented.

ADV_LLD.2.8C The low-level design shall identify the interfaces of the modules of the TSF visible at the external interface of the TSF.

ADV_LLD.2.9C The low-level design shall describe the separation of the TSF into TSP enforcing and other modules.

Evaluator action elements:

ADV_LLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.2.2E The evaluator shall determine if the functional requirements in the ST are addressed by the representation of the TSF.

ADV_RCR Representation correspondence

Objectives

814 The correspondence between the various representations (i.e. functional requirements expressed in the ST, functional specification, high-level design, low-level design, implementation) addresses the correct and complete instantiation of the requirements to the least abstract representation provided. This conclusion is

achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

Application notes

815 The developer must demonstrate to the evaluator that the most detailed, or least abstract, representation of the TSF is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

816 The evaluator must analyse each demonstration of correspondence between abstractions, as well as the results of the analysis of each TSF representation, and then make a determination as to whether the functional requirements in the ST have been satisfied.

817 This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure 5.4, it is intended to address correspondence between the requirements in the ST as well as the TOE summary specification, functional specification, high-level design, low-level design, and implementation representation.

ADV_RCR.3 Formal correspondence demonstration

Application notes

818 The developer must either demonstrate or prove correspondence, as described in the requirements below, commensurate with the level of rigour of presentation style. For example, correspondence must be proven when corresponding representations are formally specified.

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.3.1D The developer shall provide evidence that the least abstract TSF representation provided is an accurate, consistent, and complete instantiation of the functional requirements expressed in the ST.

ADV_RCR.3.2D **For those corresponding portions of representations that are formally specified, the developer shall prove that correspondence.**

Content and presentation of evidence elements:

ADV_RCR.3.1C **For each adjacent pair of TSF representations, the evidence shall prove or demonstrate that all parts of the more abstract representation are refined in the less abstract representation.**

ADV_RCR.3.2C **For each adjacent pair of TSF representations, where portions of one representation are semiformally specified and the other at least semi-formally specified, the demonstration of correspondence between those portions of the representations shall be semiformal.**

ADV_RCR.3.3C For each adjacent pair of TSF representations, where portions of either representation are informally specified the demonstration of correspondence between those portions of the representations may be informal.

ADV_RCR.3.4C **For each adjacent pair of TSF representations, where portions of both representations are formally specified the proof of correspondence between those portions of the representations shall be formal.**

Evaluator action elements:

ADV_RCR.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.3.2E The evaluator shall analyse the correspondence between the functional requirements expressed in the ST and the least abstract representation provided to ensure accuracy, consistency, and completeness.

ADV_RCR.3.3E **The evaluator shall determine the accuracy of the proofs of correspondence by selectively verifying the formal analysis.**

AGD Guidance documents

819 The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure installation and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

AGD_ADM Administrator guidance

Objectives

820 Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

Application notes

821 The requirements AGD_ADM.1.2C and AGD_ADM.1.11C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

822 The PP/ST author should review the functional components of the PP/ST for guidance on administrator documentation. Those application notes that are relevant to administrator guidance for understanding and proper application of the security functions should be considered for inclusion in the administrator guidance requirements. An example of an administrator guidance document is a reference manual.

AGD_ADM.1 Administrator guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.2C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.3C The administrator guidance shall contain guidelines on the consistent and effective use of the security functions within the TSF.

AGD_ADM.1.4C The administrator guidance shall describe the difference between two types of functions: those which allow an administrator to control security parameters, and those which allow the administrator to obtain information only.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the administrator's control.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall contain guidelines on how the security functions interact.

AGD_ADM.1.8C The administrator guidance shall contain instructions regarding how to configure the TOE.

AGD_ADM.1.9C The administrator guidance shall describe all configuration options that may be used during secure installation of the TOE.

AGD_ADM.1.10C The administrator guidance shall describe details, sufficient for use, of procedures relevant to the administration of security.

AGD_ADM.1.11C The administrator guidance shall be consistent with all other documents supplied for evaluation.

Evaluator action elements:

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_ADM.1.2E The evaluator shall confirm that the installation procedures result in a secure configuration.

AGD_USR User guidance

Objectives

823 User guidance refers to written material that is intended to be used by nonadministrative (human) users of the TOE. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

824 The user guidance provides a basis for assumptions about the use of the TOE and a measure of confidence that non-malicious users and application providers will understand the secure operation of the TOE and will use it as intended.

Application notes

825 The requirement AGD_USR.1.3.C and AGD_USR.1.5C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the user guidance.

826 The PP/ST author should review the functional components of the PP/ST for guidance on user documentation. Those application notes that are relevant to user guidance aimed at the understanding and proper use of the security functions should be considered for inclusion in the user guidance requirements. Examples of user guidance are reference manuals, user guides, and on-line help.

AGD_USR.1 User guidance

Dependencies:

ADV_FSP.1 TOE and security policy

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the TSF and interfaces available to the user.

AGD_USR.1.2C The user guidance shall contain guidelines on the use of security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall describe the interaction between user-visible security functions.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation delivered for evaluation.

Evaluator action elements:

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC Life cycle support

827 Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

ALC_DVS Development security

Objectives

828 Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

Application notes

829 The evaluator should decide whether there is a need for visiting the user's site in order to confirm that the requirements of this family are met.

ALC_DVS.2 Sufficiency of security measures

Dependencies:

No dependencies.

Developer action elements:

ALC_DVS.2.1D The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.2.1C The development security documentation shall describe the physical, procedural, personnel, and other security measures that are used to protect the confidentiality and integrity of the TOE during its development.

ALC_DVS.2.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

ALC_DVS.2.3C The evidence shall justify that the security measures are sufficient to protect the confidentiality and integrity of the TOE.

Evaluator action elements:

ALC_DVS.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.2.2E The evaluator shall check whether the security measures are being applied.

ALC_LCD Life cycle definition

Objectives

830 Poorly controlled development and maintenance can result in a flawed implementation of a TOE (or a TOE that does not meet all of its security requirements). This, in turn, results in security violations. Therefore, it is important that a model for the development and maintenance of a TOE be established as early as possible in the TOE's life-cycle.

831 Using a model for the development and maintenance of a TOE does not guarantee that the TOE will be free of flaws, nor does it guarantee that the TOE will meet all of its security functional requirements. It is possible that the model chosen was insufficient or inadequate and therefore no benefits in the quality of the TOE could be observed. Using a life-cycle model that has been approved by some group of

experts (e.g., academic experts, standards bodies) improves the chances that the development and maintenance models will contribute to the overall quality of the TOE.

Application notes

832 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming relevant after the completion of the evaluation, its evaluation adds assurance through an analysis the life-cycle information for the TOE provided at the time of the evaluation.

833 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE.

834 A standardised life-cycle model is a model that has been approved by some group of experts (e.g., academic experts, standards bodies).

835 A measurable life-cycle model is a model with some arithmetic parameters so that e.g. the coding standards can be measured.

ALC_LCD.3 Measurable life-cycle model

Dependencies:

No dependencies.

Developer action elements:

ALC_LCD.3.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.3.2D The developer shall produce life-cycle definition documentation.

ALC_LCD.3.3D **The developer shall use a standardised and measurable life-cycle model to develop and maintain the TOE.**

Content and presentation of evidence elements:

ALC_LCD.3.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.3.2C The life-cycle definition documentation shall explain why the model was chosen and how it is used to develop and maintain the TOE.

ALC_LCD.3.3C **The life-cycle definition documentation shall demonstrate compliance with the standardised and measurable life-cycle model.**

Evaluator action elements:

ALC_LCD.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT Tools and techniques

Objectives

836 Tools and techniques is an aspect of selecting tools which are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to programming languages, documentation, implementation standards, and other parts of the TOE like supporting runtime libraries.

Application notes

837 There is a requirement for well-defined development tools. These are tools which have been shown to be well understood and applicable without the need for intensive further clarification. For example, programming languages and computer aided design (CAD) systems that are based on an a standard published by standards bodies are considered to be well-defined.

838 Tools and techniques distinguishes between the implementation standards applied by the developer and the implementation standards for “all parts of the TOE” which additionally includes third party software, hardware, or firmware.

839 The requirement in ALC_TAT.1.2C is specifically applicable to programming languages so as to ensure that all statements in the source code have an unambiguous meaning.

ALC_TAT.3 Compliance with implementation standards - all parts

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements:

ALC_TAT.3.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.3.2D The developer shall document the selected implementation dependent options of the development tools.

ALC_TAT.3.3D The developer shall describe the implementation standards for all parts of the TOE.

Content and presentation of evidence elements:

ALC_TAT.3.1C Any development tools used for implementation shall be well-defined.

ALC_TAT.3.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

Evaluator action elements:

ALC_TAT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.3.2E The evaluator shall confirm that the implementation standards have been applied.

ATE Tests

840 The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g., functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of the PP/ ST. Testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. Testing may also be directed toward the internals of the TSF, such as the testing of subsystems and modules against their specifications.

841 The aspects of coverage and depth have been separated from functional tests for reasons of increased flexibility in applying the components of the families. However, the requirements in these three families are intended to be applied together.

842 The independent testing has dependencies on the other families to provide the necessary information to support the requirements, but is primarily concerned with independent evaluator actions.

843 This class does not address penetration testing, which is directed toward finding vulnerabilities that enable a user to violate the security policy. Penetration testing is addressed separately as an aspect of vulnerability assessment in the class AVA.

ATE_COV Coverage

Objectives

844 This family addresses those aspects of testing that deal with completeness of testing. That is, it addresses the extent to which the TOE security functions are tested, whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified, and whether or not the order in which testing proceeds

correctly accounts for functional dependencies between the portions of the TOE being tested.

Application notes

845 The specific documentation required by the coverage components will be determined, in most cases, by the documentation stipulated in the level of ATE_FUN that is specified. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_COV.3 Ordered testing

Objectives

846 The objective is that testing completely address the security functions.

847 The objective is to ensure that there is a detailed correspondence between the tests and the security functions.

848 In this component, an additional objective is detailed justification that testing is structured such as to avoid circular arguments about the correctness of the portions of the TOE being tested.

Application notes

849 Ordering dependencies between tests can be of different forms e.g., test A provides a result to test B; test A cannot run before test B, since it breaks something required by test B; test failure in test B might be because of a failure in “untested” test A.

Dependencies:

ADV_FSP.1 TOE and security policy

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.3.1D The developer shall provide an analysis of the test coverage.

ATE_COV.3.2D The developer shall provide an analysis of ordering dependencies of tests.

Content and presentation of evidence elements:

ATE_COV.3.1C The analysis of the test coverage shall demonstrate that the tests identified in the test documentation cover the TSF.

ATE_COV.3.2C The analysis of the test coverage shall demonstrate the correspondence between the security functions and the tests identified in the test documentation.

ATE_COV.3.3C The analysis documentation shall justify that the correspondence is complete.

ATE_COV.3.4C The analysis documentation shall describe the ordering dependencies of tests.

ATE_COV.3.5C The analysis documentation shall justify that the test plans and procedures are consistent with the ordering dependencies of tests.

Evaluator action elements:

ATE_COV.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT Depth

Objectives

850 The components in this family deal with the level of detail to which the TOE is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.

851 The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internals of the TOE, are more likely to discover any malicious code that has been inserted.

Application notes

852 The specific amount and type of documentation and evidence will, in general, be determined by that required by level of ATE_FUN selected. However, the PP/ST author will need to give consideration to the proper set of test evidence and documentation required.

ATE_DPT.4 Testing - implementation

Objectives

853 The functional specification of a TOE provides a high level description of the external workings of the TSF. Testing at the level of the functional specification, in order to demonstrate the presence of any flaws, provides assurance that the TSF functional specification has been correctly realised.

854 The subsystems of a TOE provide a high level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

855 The modules of a TOE provide a description of the internal workings of the TSF. Testing at the level of the modules, in order to demonstrate the presence of any flaws, provides assurance that the TSF modules have been correctly realised.

856 The implementation representation of a TOE provides a detailed description of the internal workings of the TSF. Testing at the level of the implementation, in order to demonstrate the presence of any flaws, provides assurance that the TSF implementation has been correctly realised.

Application notes

857 The functional specification representation is used to express the notion of the most abstract representation of the TSF.

858 The developer is expected to describe the testing of the high level design of the TSF in terms of “subsystems”. The term “subsystem” is used to express the notion of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar notion of decomposition.

859 The developer is expected to describe the testing of the low level design of the TSF in terms of “modules”. The term “modules” is used to express the notion of decomposing each of the “subsystems” of the TSF into a relatively small number of parts. While the developer is not required to actually have “modules”, the developer is expected to represent a similar notion of decomposition.

860 The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one which is used to generate the TSF itself (e.g., source code which is then compiled).

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ADV_IMP.2 Implementation of the TSF

ADV_LLD.1 Descriptive low-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.4.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.4.1C **The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TOE operates in accordance with the functional specification, high level design, low level design, and implementation of the TSF.**

Evaluator action elements:

ATE_DPT.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_FUN Functional tests

Objectives

861 Functional testing establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family “Functional tests” is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through testing.

862 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

Application notes

863 Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures should also show how the test results is derived from the test inputs.

864 The developer shall eliminate all security relevant flaws discovered during testing.

865 The developer shall test the TSF to determine that no new security relevant flaws have been introduced as a result of eliminating discovered security relevant flaws.

ATE_FUN.1 Functional testing

Objectives

866 The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

Dependencies:

ATE_COV.1 Complete coverage - informal

ATE_DPT.1 Testing - functional specification

Developer action elements:

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, and test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function.

ATE_FUN.1.4C The test results in the test documentation shall show the expected results of each test.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each security function operates as specified.

Evaluator action elements:

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND Independent testing

Objectives

867 The objective is to demonstrate that the security functions perform as specified.

868 Additionally, an objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer which results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

Application notes

869 The testing specified in this family can be performed by a party other than the evaluator (e.g., an independent laboratory, an objective consumer organisation).

870 This family deals with the degree to which there is independent functional testing of the TOE. Independent functional testing may take the form of repeating the developer's functional tests, in whole or in part. It may also take the form of the augmentation of the developer's functional tests, either to extend the scope or the depth of the developer's tests.

ATE_IND.3 Independent testing - complete

Objectives

871 The objective is to demonstrate that all security functions perform as specified.

872 In this component, the objective is to repeat the developer testing.

Application notes

873 The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required to run tests. The need for documentation is supported by the dependencies to other assurance families.

874 Additionally, suitability of the TOE for testing may be based on other considerations e.g., the version of the TOE submitted by the developer is not the final version.

875 The developer is required to perform testing and to provide test documentation and test results. This is addressed by the ATE_FUN family.

Dependencies:

ADV_FSP.1 TOE and security policy

AGD_USR.1 User guidance

AGD_ADM.1 Administrator guidance

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.3.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.3.1C The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.3.2E The evaluator shall test the TSF to confirm that the TSF operates as specified.

ATE_IND.3.3E **The evaluator shall execute all tests in the test documentation to verify the developer test results.**

AVA Vulnerability assessment

876 The class “Vulnerability assessment” encompasses four families: covert channel analysis (AVA_CCA), misuse (AVA_MSU), strength of TOE security functions (AVA_SOF) and vulnerability analysis (AVA_VLA). The class addresses the existence of exploitable covert channels, the misuse or incorrect configuration of the TOE, the ability for all critical security mechanisms to withstand direct attack and the definition and assessment of penetration tests to exploit vulnerabilities introduced in the development or the operation of the TOE.

AVA_CCA Covert channel analysis

Objectives

877 Covert channel analysis is carried out to determine the existence and potential capacity of unintended signalling channels that may be exploited by malicious code.

878 The assurance requirements address the threat that unintended and exploitable signalling paths exist which may be exercised to violate the security policy.

Application notes

879 Channel capacity estimations are based upon informal engineering measurements, as well as actual test measurements.

880 Details of the assumptions upon which the covert channel analysis is based shall be given, e.g., processor speed, configuration, memory, and cache size.

881 Test parameters details are (e.g., processor speed, memory and cache size), relevant configuration parameters, how the channel was exercised, used to obtain the capacity during testing.

882 The selective validation of the covert channel analysis through testing allows the evaluator the opportunity to verify any aspect of the covert channel analysis (e.g., identification, capacity estimation, elimination, monitoring, and exploitation scenarios). This does not impose a requirement to demonstrate the entire set of covert channel analysis results.

883 If there are no information flow control policies in the ST, this family of assurance requirements is no longer applicable since this family only applies to information flow control policies. Even if there are no specific functional requirements (e.g., FDP_IFF.1 to FDP_IFF.3) for eliminating, limiting, or monitoring covert channels, this family still requires the identification of covert channels.

AVA_CCA.2 Systematic covert channel analysis

Objectives

884 The objective is to identify covert channels which are identifiable through analysis.

885 In this component, the objective is to perform a systematic search for covert channels.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_IMP.2 Implementation of the TSF

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_CCA.2.1D The developer shall conduct a search for covert channels for each information flow control policy.

AVA_CCA.2.2D The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements:

AVA_CCA.2.1C The analysis documentation shall identify covert channels.

AVA_CCA.2.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA.2.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA.2.4C The analysis documentation shall describe the method used for estimating channel capacity, which shall be based on worst case scenarios.

AVA_CCA.2.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_CCA.2.6C The analysis documentation shall provide evidence that the method used to identify covert channels is systematic.

Evaluator action elements:

AVA_CCA.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA.2.2E The evaluator shall confirm that the results of the covert channels analysis meet the functional requirements.

AVA_CCA.2.3E The evaluator shall selectively validate the covert channel analysis through testing.

AVA_MSU Misuse

Objectives

886 Misuse investigates whether the TOE can be configured or used in a manner which is insecure but which an administrator or end-user of the TOE would reasonably believe to be secure.

887 The objective is to minimise the risk of human or other errors in operation which may deactivate, disable, or fail to activate security functions.

888 The objective is to minimise the probability of configuring or installing the TOE in a way which is insecure, without the end user or administrator being able to recognise it.

Application notes

889 Conflicting, misleading or incomplete guidance may result in a user of the TOE believing that the TOE is secure, when it is not. Conflicting guidance can result in vulnerabilities.

890 An example of conflicting guidance would be two guidance instructions which imply different outcomes when the same input is supplied.

891 An example of misleading guidance would be the description of a single guidance instruction which could be parsed in more than one way, one of which may result in an insecure state.

892 An example of completeness would be referencing assertions of dependencies on external security measures e.g., such as external procedural, physical and personnel controls.

AVA_MSU.2 Misuse analysis - independent verification

Objectives

893 The objective is to ensure that conflicting guidance in the guidance documentation have been addressed.

894 In this component, the objective is to provide additional assurance by performing an independent analysis.

Dependencies:

ADO_IGS.1 Installation, generation, and start-up procedures

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.2.1D The developer shall document an analysis of the guidance documentation for conflicting and incomplete guidance.

AVA_MSU.2.2D The developer shall ensure that the guidance documentation contains no misleading or unreasonable guidance.

Content and presentation of evidence elements:

AVA_MSU.2.1C The analysis documentation shall provide a rationale that demonstrates that the guidance is not conflicting and is complete.

Evaluator action elements:

AVA_MSU.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E The evaluator shall determine that there is no misleading or unreasonable guidance in the guidance documentation.

AVA_MSU.2.3E The evaluator shall repeat any procedures in the guidance documentation to ensure that they produce the documented results.

AVA_MSU.2.4E The evaluator shall perform independent testing to confirm that the TOE can be configured and operated securely using only the guidance documentation.

AVA_SOF Strength of TOE security functions

Objectives

895 Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security functions claim.

Application notes

896 Security functions are implemented by security mechanisms. For example, a password mechanism can be used in the implementation of the identification and authentication security function.

897 The strength of TOE security functions evaluation is performed at the level of the security mechanism, but its results provide knowledge about the ability of the related security function to counter the identified threats.

898 The strength of a function is rated ‘basic’ if the analysis shows that the function provides adequate protection against unintended or casual breach of TOE security by attackers possessing a low attack potential.

899 The strength of a function is rated ‘medium’ if the analysis shows that the function provides adequate protection against attackers possessing a moderate attack potential.

900 The strength of a function is rated ‘high’ if the analysis shows that the function provides adequate protection against attackers possessing a high attack potential.

901 The attack potential is derived from the attacker’s expertise, opportunities, resources, and motivation.

AVA_SOF.1 Strength of TOE security function evaluation

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

Developer action elements:

AVA_SOF.1.1D The developer shall identify all TOE security mechanisms for which a strength of TOE security function analysis is appropriate.

AVA_SOF.1.2D The developer shall perform a strength of TOE security function analysis for each identified mechanism.

Content and presentation of evidence elements:

AVA_SOF.1.1C The strength of TOE security function analysis shall determine the impact of the identified TOE security mechanisms on the ability of the TOE security functions to counter the threats.

AVA_SOF.1.2C The strength of TOE security function analysis shall demonstrate that the identified strength of the security functions is consistent with the security objectives of the TOE.

AVA_SOF.1.3C Each strength claim shall be either basic, medium, or high.

Evaluator action elements:

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that all TOE security mechanisms requiring a strength analysis have been identified.

AVA_SOF.1.3E The evaluator shall confirm that the strength claims are correct.

AVA_VLA Vulnerability analysis

Objectives

902 Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or e.g., by flaw hypotheses, could allow malicious users to violate the TSP.

903 Vulnerability analysis deals with the threats that a malicious user will be able to discover flaws that will allow access to resources (e.g., data), allow the ability to interfere with or alter the TSF, or interfere with the authorised capabilities of other users.

Application notes

904 The vulnerability analysis should consider the contents of all the TOE deliverables for the targeted evaluation assurance level.

905 Obvious vulnerabilities are those that allow common attacks or those that might be suggested by the TOE interface description. Obvious vulnerabilities are those in the public domain, details of which should be known to a developer or available from an evaluation oversight body.

906 The evidence identifies all the TOE documentation upon which the search for flaws was based.

AVA_VLA.4 Highly resistant

Objectives

907 A vulnerability analysis is performed by the developer to ascertain the presence of "obvious" security vulnerabilities.

908 The objective is to confirm that no identified security vulnerabilities can be exploited in the intended environment for the TOE.

909 An independent vulnerability analysis is performed by the evaluator, which goes beyond the "obvious" security vulnerabilities. The analysis considers the deliverables available for the targeted evaluation assurance level.

910 In addition, the independent vulnerability analysis performed by the evaluator is based on analytical techniques which are employed to discover vulnerabilities that would require sophisticated attackers.

911 The TOE must be shown to be highly resistant to penetration attacks.

Application notes

912 Obvious vulnerabilities are those which are open to exploitation which requires a minimum of understanding of the TOE, skill, technical sophistication, and resources.

913 Independent vulnerability analysis is based on highly detailed technical information. The attacker is assumed to be thoroughly familiar with the specific implementation of the TOE. The attacker is presumed to have a high level of technical sophistication.

Dependencies:

ADV_FSP.1 TOE and security policy

ADV_HLD.1 Descriptive high-level design

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.4.1D The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.

AVA_VLA.4.2D The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.4.1C The evidence shall show, for each vulnerability, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.4.2C The documentation shall justify that the TOE, with the identified vulnerabilities, is highly resistant to penetration attacks.

AVA_VLA.4.3C The analysis documentation shall provide a justification that the analysis completely addresses the TOE deliverables.

Evaluator action elements:

AVA_VLA.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.4.2E The evaluator shall conduct penetration testing, based on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.

AVA_VLA.4.3E The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.4.4E The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of identified vulnerabilities in the target environment.

AVA_VLA.4.5E The evaluator shall determine that the TOE is highly resistant to penetration attacks.